TUGAS AKHIR (NA 1701)

KOMPUTERISASI PERHITUNGAN DISTRIBUSI BEBAN

PADA FLOATING DOK PARE-PARE UNTUK MENCAPAI DEFLEKSI MINIMUM

Disusun oleh
IRWIN SRIWANTO
NRP : 4195 100 505

JURUSAN TEKNIK PERKAPALAN
FAKULTAS TEKNOLOGI KELAUTAN
INSTITUT TEKNOLOGI SEPULUH NOVEMBER
SURABAYA
1999

JURUSAN TEKNIK PERKAPALAN

FAKULTAS TEKNOLOGI KELAUTAN ITS

SURAT KEPUTUSAN TUGAS AKHIR (NA 1701)

No. : 143 /PT12.FTK2/M/199 7

Nama Mahasiswa	:	Irwin Sriwanto
Nomor Pokok	:	4195100505
Tanggal diberikan tugas	:	01 . Nopember . 1997
Tanggal selesai tugas	:	15 . Pebruari . 1998
Dosen Pembimbing	:	1. Ir. Petrus. Eko. Panunggal,. Ph.D.
		2. .....

Uraian / judul tugas akhir yang diberikan :

KOMPUTERISASI PERHITUNGAN DISTRIBUSI BEBAN PADA FLOATING DOCK PARE-PARE UNTUK-MENCAPAI DEFLEKSI MINIMUM

sOn

Surabaya, 24 Nopember 1997
Jurusan Teknik Perkapalan FTK-ITS
K e t u a,

Ir. Kesstowo Sastro Wiyono.
NIP. 130 687 430.

Tembusan :
1. Yth. Dekan FTK-ITS.
2. Yth. Dosen Pembimbing.
3. Arsip.

LEMBAR PENGESAHAN

KOMPUTERISASI PERHITUNGAN DISTRIBUSI BEBAN PADA FLOATING DOK PARE-PARE UNTUK MENCAPAI DEFLEKSI MINIMUM

Menyetujui

Dosen Pembimbing

15/2 99

Ir. P. Eko Panunggal, PhD

N I P : 130 286 963

LEMBAR PENGESAHAN

(Setelah direvisi sesuai proses verbal Tugas Akhir)

Nama	:	**IRWIN SRIWANTO**
NRP	:	**4195 100 505**
Fak / Jur.	:	**FTK / T. Perkapalan**
Judul TA	:	**Komputerisasi Perhitungan Distribusi Beban Pada Floating Dok Pare-Pare Untuk Mencapai Defleksi Minimum**

DEPARTEMEN PENDIDIKAN DAN KEBUDAYAAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI KELAUTAN
JURUSAN TEKNIK PERKAPALAN

Surabaya, 13 AGUSTUS 1999

Mengetahui / Menyetujui

Dosen Pembimbing

Ir. P. Eko Panunggal, PhD

N I P : 130 286 963

ABSTRAK

Dalam suatu konstruksi yang mengalami beban lateral akan mengakibatkan terjadinya defleksi, bila defleksi ini melebihi batas yang diijinkan maka akan terjadi sesuatu yang tidak kita inginkan. Begitu pula halnya dengan defleksi pada floating dock dan kapal. Dengan menggunakan metode optimasi (volume tangki ballast sebagai variabel) kita dapat mengestimasi besarnya defleksi yang terjadi akibat beban-beban tersebut.

Dalam perhitungan beban floating dock dan kapal yang naik dok bila defleksi keel block diabaikan (dianggap sama) maka defleksi floating dock dan kapal akan sama (selisih = 0). Apabila beban yang terdiri dari berat dock, ballast, buoyancy dan berat kapal, kita anggap tetap maka reaksi keel block akan menjadi variabel untuk mendapatkan w dock dan w kapal sama. Reaksi keel block diperoleh dengan menggunakan metode Newton Raphson dengan variabel banyak. Untuk mendapatkan harga minimum dari defleksi digunakan metode optimasi yaitu dengan metode anealing.

Dari hasil program didapatkan besarnya harga defleksi minimum adalah sekitar 0.03 mm. Harga ini jauh lebih kecil dari batas yang diijinkan untuk defleksi pada floating dock yaitu sekitar 100 mm sehingga defleksi yang terjadi sangat kecil.

KATA PENGANTAR

Dengan mengucapkan puji syukur kepada Allah swt, akhirnya tugas akhir ini dapat diselesaikan meskipun tidak sedikit hambatan yang ditemui. Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar sarjana di Jurusan Teknik Perkapalan Fakultas Teknologi Kelautan Institut Teknologi Sepuluh November Surabaya.

Selain itu penulis mengucapkan rasa terima kasih yang sebesar-besarnya kepada :

- Ayah, Ibu dan kedua adikku yang telah memberikan dorongan moril.
- Bapak Ir. P. Eko Panunggal, PhD selaku dosen pembimbing.
- Bapak Ir. Koestowo selaku ketua Jurusan Teknik Perkapalan.
- Rekan-rekan lintas jalur 1995 Jurusan Teknik Perkapalan.
- Rekan-rekan reguler Jurusan Teknik Perkapalan.

Seperti kata pepatah 'tiada gading yang tak retak' maka tugas akhir ini masih jauh dari sempurna sehingga saran, kritik dan perbaikan sangatlah diharapkan dari para pembaca. Semoga tugas akhir ini dapat bermanfaat bagi rekan-rekan lainnya.

Surabaya, Juli 1999

Penulis

DAFTAR ISI

BAB 1

PENDAHULUAN

1.1. LATAR BELAKANG

Dalam proses pengedokan kapal di floating dock khususnya di floating dock di PT. PAL Surabaya sampai dengan saat ini belum ada prosedur yang mengatur aspek-aspek teknis secara benar.

Pelaksanaan docking kapal hanya berpedoman pada pengalaman dan prinsip mempertahankan kerataan (dalam hal ini trim) dalam proses pengangkutan kapal (pemompaan).

Seperti diketahui dalam proses pengedokan kapal akan terjadi distribusi beban yang diterima dok berasal dari beban kapal dan gaya buoyancy pada waktu pemompaan maupun sebagai fungsi benda terapung.

Pada Tugas Akhir karya Yani Rusihadi telah dilakukan pembuatan program untuk menghitung reaksi keel block dengan memberikan harga awal dari beban-beban yang bekerja pada floating dock. Program tersebut memberikan hasil atau output berupa harga defleksi (Rusihadi, 1997).

Dari hasil tersebut timbul pertanyaan apakah harga defleksi tersebut sudah minimum. Untuk memperoleh harga defleksi yang benar-benar minimum maka dicoba untuk menerapkan metode optimasi. Dalam metode ini yang berfungsi sebagai variabel adalah volume tangki ballast.

Dengan penggunaan metode optimasi diharapkan harga defleksi yang dihasilkan nantinya benar-benar minimum. Dan juga volume tiap tangki ballast dapat diketahui harganya sehingga dapat dihindarkan hal-hal yang tidak diinginkan seperti terjadinya buckling pada floating dock.

1.2. PERUMUSAN MASALAH

Harga defleksi yang dihasilkan oleh program pada Tugas Akhir karya Yani Rusihadi tidak dapat dikatakan sebagai harga minimum. Selain itu apabila jumlah keel block ditambah maka kecepatan dari metode invers yang ada yaitu metode Gauss Jordan berkurang.

Oleh karena itu dalam Tugas Akhir ini akan dibahas mengenai cara untuk mengatasi kedua masalah tersebut diatas yaitu :

- ❖ Pemilihan metode invers yang lebih baik sehingga kecepatan dari invers tidak berkurang meskipun jumlah keel block dinaikkan mendekati jumlah sebenarnya.
- ❖ Penggunaan metode optimasi untuk mengatur distribusi volume tangki ballast agar defleksi yang terjadi menjadi seminim mungkin.

1.3. TUJUAN

Tujuan dari penulisan Tugas Akhir ini adalah :

- Agar tidak terjadi kerusakan pada floating dok dan kapal yang naik dok
- Agar floating dok tidak mengalami buckling
- Agar dapat memperpanjang umur pakai dari floating dok

- Dapat digunakan sebagai guidance atau petunjuk bagi operator dok dalam pengoperasian floating dok

1.4. METODOLOGI PENELITIAN

Prosedur yang akan dilakukan adalah sebagai berikut :

- Input data berupa harga beban-beban yang bekerja pada floating dock dan kapal.
- Melakukan perhitungan invers matriks beban dengan menggunakan metode LU decomposition. Dibandingkan dengan metode Gauss-Jordan, metode ini memerlukan 1/3 N^3 eksekusi inner loop dan mempunyai faktor kecepatan 3 kali lebih baik. Selain itu kekurangan pada Gauss-Jordan adalah semua komponen di ruas kanan harus diketahui terlebih dahulu (Al-Khafaj, 1986).
- Perhitungan reaksi keel blok baru.
- Pengecekan harga reaksi keel blok dengan batasan ketelitian yang telah ditentukan.
- Penggunaan metode optimasi dalam hal ini metode Anealing untuk mencari harga defleksi minimum dengan volume tangki ballast sebagai variabel. Karena yang akan diminimumkan adalah defleksi maka fungsi obyektif adalah jumlah kuadrat defleksi.
- Output dari proses optimasi yaitu harga defleksi dicek apakah sudah memenuhi syarat optimum. Adapun syarat optimum adalah apabila fungsi obyektif kurang dari nol atau dengan probabilitas exp(-de/t) dimana de adalah

fungsi obyektif dan t adalah parameter suhu yang harganya telah ditentukan terlebih dahulu (Press, 1989).

- Jika memenuhi maka langkah tersebut dicatat sebagai langkah yang berhasil (mencapai optimum) dan dilakukan juga pengacakan harga volume tangki ballast. Hal ini dilakukan untuk memperoleh volume tangki ballast yang berbeda.
- Proses perhitungan diulangi lagi dari awal hingga batas iterasi yang telah ditentukan misalnya 100 iterasi. Atau iterasi dihentikan bila jumlah langkah yang berhasil (mencapai optimum) telah mencapai batas yang ditentukan misalnya 50 langkah.

1.5. BATASAN MASALAH

Pada Tugas Akhir ini permasalahan dibatasi pada :

- Defleksi kapal dan floating dok dianggap sama
- Jumlah keel blok sebanyak 20 buah
- Variabel masukan untuk metode optimasi adalah distribusi tangki ballast
- Pengisian tangki ballast kiri dan kanan adalah sama
- Panjang dan lebar tangki ballast adalah tetap.

BAB 2

DASAR TEORI

2.1. PENDAHULUAN

Sebuah balok akan mengalami defleksi dari kedudukannya semula bila berada dibawah pengaruh gaya yang bekerja pada balok tersebut.

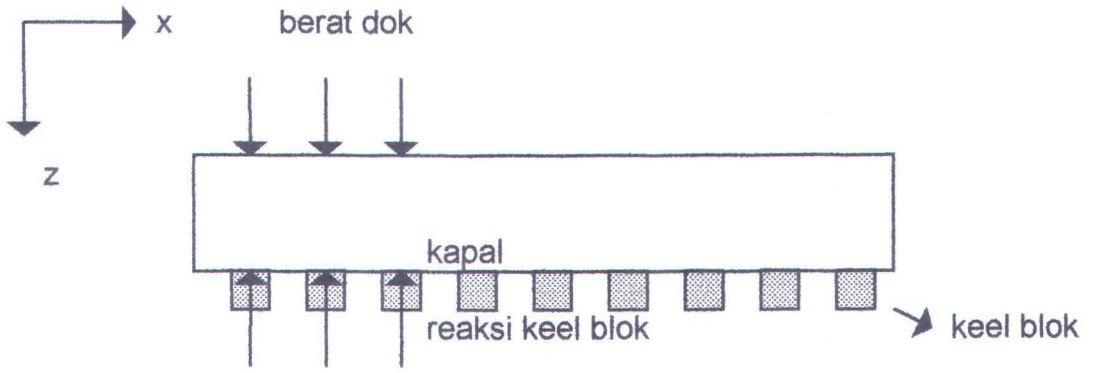
Sebagai pendekatan awal untuk menentukan besar dari gaya geser dan momen yang terjadi pada badan kapal diasumsikan bahwa kapal dan floating dok sebagai balok yang elastis. Sedang suatu konstruksi dapat disebut balok jika pada konstruksi tersebut hanya menerima beban tegak lurus dengan garis sumbu konstruksi tersebut atau dapat dikatakan beban yang bekerja adalah beban lateral. Untuk memenuhi persamaan kompatibilitas diasumsikan bahwa bidang penampang melintang balok akan tetap datar dan tegak lurus dengan sumbu balok setelah deformasi (Popov, 1989).

2.1.1. PEMODELAN

Pemodelan yang dipakai sama dengan pemodelan yang digunakan dalam Tugas Akhir karya Yani Rusihadi (Rusihadi, 1997).

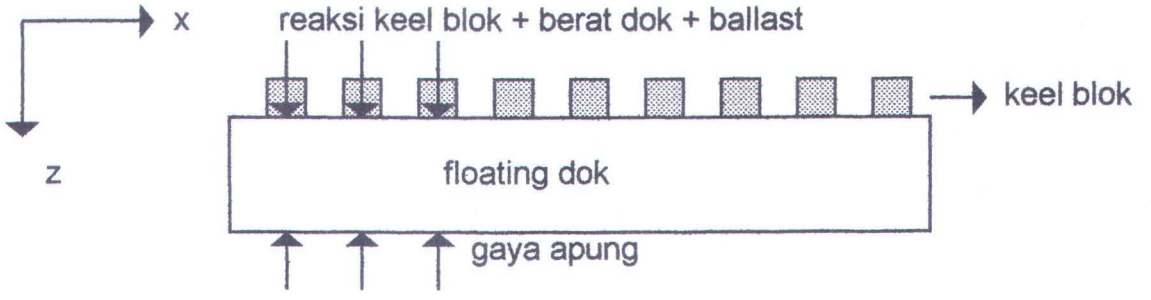
Model Pembebanan Kapal

Beban pada kapal = beban kapal sendiri - reaksi keel blok

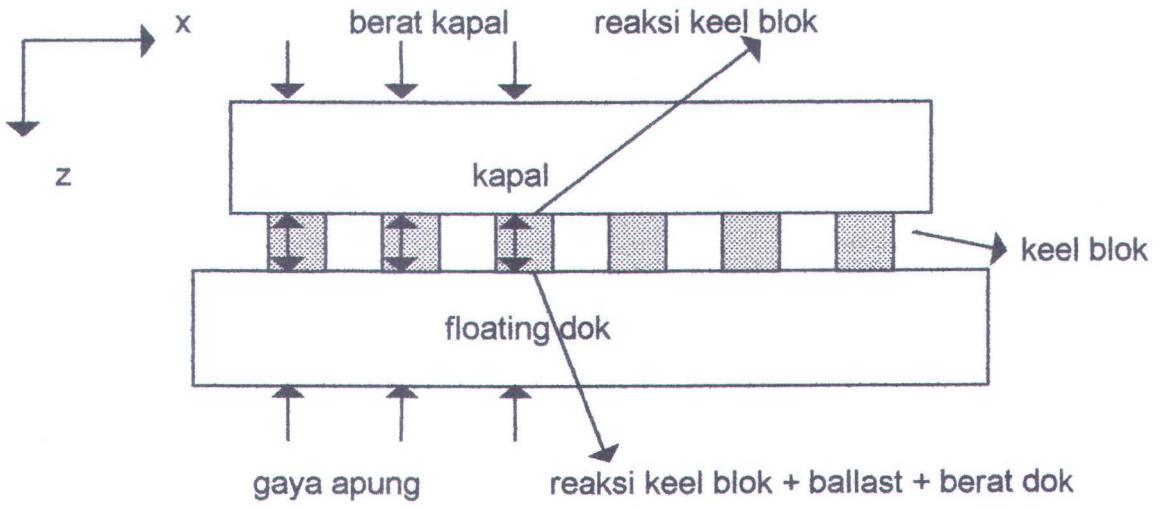


Gambar Model Pembebanan Kapal

Beban floating = gaya apung - (reaksi keel blok + berat ballast + berat dok)



Gambar Model Pembebanan Floating Dock



Gambar Pembebanan Gabungan

Dari grafik diatas akan diperoleh dua gaya yaitu gaya yang bekerja pada kapal q_{kapal} dan gaya yang bekerja pada floating dock q_{dock}. Sama seperti mencari harga defleksi pada balok maka gaya lintang didapatkan setelah dilakukan integral pertama terhadap beban-beban tersebut.

Kemudian untuk mendapatkan momen lengkung dilakukan integral terhadap gaya lintang. Selanjutnya harga sudut lentur (slope) diperoleh setelah dilakukan integral terhadap momen lengkung. Harga defleksi kita peroleh setelah dilaksanakan integral terhadap slope.

Karena pada floating dock yang berfungsi sebagai penumpu kapal adalah keel block maka beban yang bekerja tersebar pada keel block tersebut. Sehingga persamaan yang ada mempunyai banyak variabel. Salah satu cara untuk menyelesaikan persamaan jenis ini adalah dengan metode Newton Raphson.

2.1.2. METODE NEWTON RAPHSON

Pada batasan masalah diatas disebutkan bahwa keel block dianggap tidak mengalami defleksi, sehingga defleksi dok dan defleksi kapal harus sama. Agar harga kedua defleksi tersebut sama maka harus dihitung terlebih dahulu harga gaya reaksi keel block tiap station.

Salah satu metode untuk mencari harga variabel (reaksi keel block) yang banyak dengan dua persamaan atau lebih adalah menggunakan metode Newton Raphson dengan variabel banyak.

Dalam (Al-Khafaj, 1986) diterangkan :

$$\begin{bmatrix} \frac{df_1(x_1)}{dx_1} & \frac{df_1(x_2)}{dx_2} & \cdots & \frac{df1(x_k)}{dx_k} \\ \frac{df_2(x_1)}{dx_1} & \frac{df_2(x_2)}{dx_2} & \cdots & \frac{df2(x_k)}{dx_k} \\ \frac{df_3(x_1)}{dx_1} & \frac{df_3(x_2)}{dx_2} & \cdots & \frac{df3(x_k)}{dx_k} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{dfi(x_1)}{dx_1} & \frac{dfi(x_2)}{dx_2} & \cdots & \frac{dfi(x_k)}{dx_k} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdots \\ h_k \end{bmatrix} = - \begin{bmatrix} f_1(x_i) \\ f_2(x_i) \\ \cdots \\ f_n(x_i) \end{bmatrix}$$

{ h } adalah vektor increment dan { f } adalah matriks fungsi.

$\frac{df1(x4)}{dx4}$ artinya defleksi 1 (kesatu) akibat perubahan harga reaksi keel blok ke 4 (empat) yaitu harga reaksi keel blok $x_4 + \Delta x_4$, dimana harga $\Delta x_4 = x_4 * 10^{-7}$. Sedangkan harga reaksi keel blok lainnya tetap.

Jadi dengan mengadakan perubahan satu reaksi keel blok kita akan mendapatkan satu kolom harga matrik Jacobian.

Dalam bentuk persamaan matematis dapat ditulis sebagai :

$$[J]\{h\} = \{f\} \qquad \ldots\ldots(1)$$

$$\{h\} = -[J]^{-1}\{f\}$$

dimana : $h_1 = \bar{x}_1 - x_1$

$h_2 = \bar{x}_2 - x_2$

$h_n = \bar{x}_n - x_n$

maka persamaan diatas dapat ditulis ulang menjadi :

$$\{\bar{x}\} = \{x\} - [J]^{-1}\{f\}$$

Kemudian dilakukan iterasi sehingga memenuhi kriteria dalam batas toleransi yang telah ditetapkan yaitu :

$$T \geq \sqrt{(h_1)^2 + (h_2)^2 + \ldots + (h_n)^2}$$

$$T \geq \sqrt{\sum_{i=1}^{n} h_i^2}$$

2.2. METODE INVERS

Untuk menyelesaikan persamaan (1) diatas, kita harus mengetahui invers matrik. Ada beberapa metode untuk mendapatkan harga invers matrik. Dalam Tugas Akhir ini digunakan metode LU decomposition yang mempunyai beberapa kelebihan dibandingkan metode Gauss Jordan yang digunakan pada Tugas Akhir karya Yani Rusihadi. Untuk memperoleh perbandingan antara metode invers berikut akan dijelaskan secara umum mengenai beberapa metode invers serta kekurangan dan kelebihan dari metode tersebut.

2.2.1. Aturan Cramer

Meskipun metode ini kurang efisien dalam menyelesaikan persamaan linear homogen dalam bentuk banyak tetapi dapat digunakan untuk menjelaskan masalah yang berkaitan dengan penyelesaian persamaan aljabar linear (Al-Khafaj, 1986). Misalnya diberikan dua persamaan tidak homogen seperti dibawah ini :

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \begin{Bmatrix} x1 \\ x2 \end{Bmatrix} = \begin{Bmatrix} b1 \\ b2 \end{Bmatrix}$$

Perkalian bentuk diatas dengan transpose matrik dari kofaktor matrik koefisien menghasilkan :

$$\begin{bmatrix} a22 & -a12 \\ -a21 & a11 \end{bmatrix} \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \begin{Bmatrix} x1 \\ x2 \end{Bmatrix} = \begin{bmatrix} a22 & -a12 \\ -a21 & a11 \end{bmatrix} \begin{Bmatrix} b1 \\ b2 \end{Bmatrix}$$

yang dapat disederhanakan menjadi :

$$\begin{bmatrix} a11a22 - a12a21 & 0 \\ 0 & a11a22 - a12a21 \end{bmatrix} \begin{Bmatrix} x1 \\ x2 \end{Bmatrix} = \begin{Bmatrix} b1 \\ b2 \end{Bmatrix}$$

$$x_1 = \frac{a22b1 - a12b2}{a11a22 - a12a21} = \frac{\begin{vmatrix} b1 & a12 \\ b2 & a22 \end{vmatrix}}{\begin{vmatrix} a11 & a12 \\ a21 & a22 \end{vmatrix}}$$

$$x_2 = \frac{a11b2 - a21b1}{a11a22 - a12a21} = \frac{\begin{vmatrix} a11 & b1 \\ a21 & b2 \end{vmatrix}}{\begin{vmatrix} a11 & a12 \\ a21 & a22 \end{vmatrix}}$$

Secara umum dapat ditulis : $[a_{ij}] \{ x_j \} = \{ b_j \}$

dimana i = 1,...,n dan j = 1,....,n

$$x_j = \frac{|Aj|}{|A|}$$

2.2.2. Eliminasi Gauss

Merupakan metode yang paling populer dan efisien dalam menyelesaikan n x n sistem persamaan (Al-Khafaj, 1986).

Diberikan contoh soal yaitu :

$2x_2 + 4x_3 = 6$

$4x_1 + x_2 - 3x_3 = 1$

$3x_1 - 8x_2 + 2x_3 = 2$

Persamaan diatas disusun menjadi bentuk matrik dimana $a_{11} = 1$

$$\left[\begin{array}{ccc|c} 4 & 1 & -3 & 1 \\ 0 & 2 & 4 & 6 \\ 3 & -8 & 2 & 2 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} 1 & 0.25 & -0.75 & 0.25 \\ 0 & 2 & 4 & 6 \\ 3 & -8 & 2 & 2 \end{array}\right] \quad R1/4$$

$$\left[\begin{array}{ccc|c} 1 & 0.25 & -0.75 & 0.25 \\ 0 & 2 & 4 & 6 \\ 0 & -8.75 & 4.25 & 1.25 \end{array}\right] \quad -3R1 + R3$$

$$\left[\begin{array}{ccc|c} 1 & 0.25 & -0.75 & 0.25 \\ 0 & 1 & 2 & 3 \\ 0 & -8.75 & 4.25 & 1.25 \end{array}\right] \quad R2/2$$

$$\left[\begin{array}{ccc|c} 1 & 0.25 & -0.75 & 0.25 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 21.75 & 27.5 \end{array}\right] \quad 8.75R2 + R3$$

Dalam bentuk persamaan aljabar :

$x_1 + 0.25\, x_2 - 0.75\, x_3 = 0.25$

$x_2 + 2\, x_3 = 3$

$x_3 = 1.264$

Dengan cara substitusi didapat hasil :

$x_3 = 1.264$

$x_2 = 0.472$ $x_1 = 1.080$

2.2.3. Eliminasi Gauss-Jordan

Metode ini merupakan pengembangan dari metode Gauss. Perbedaan yang utama adalah tidak diperlukan substitusi.

Dengan tidak adanya substitusi maka waktu yang diperlukan untuk memperoleh invers akan lebih singkat (lihat Al-Khafaj, 1986).

2.2.4. LU Decomposition

Misalnya diberikan matrik A sebagai berikut :

$$\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix}$$

Matrik tersebut diatas dapat ditulis sebagai produk dari dua matrik yaitu :

$$L \cdot U = A$$

dimana L adalah triangular bawah (hanya mempunyai elemen pada diagonal dan dibawah diagonal) dan U adalah triangular atas (hanya mempunyai elemen diagonal dan diatas diagonal).

$$L = \begin{bmatrix} \alpha 11 & 0 & 0 & 0 \\ \alpha 21 & \alpha 22 & 0 & 0 \\ \alpha 31 & \alpha 32 & \alpha 33 & 0 \\ \alpha 41 & \alpha 42 & \alpha 43 & \alpha 44 \end{bmatrix}$$

$$U = \begin{bmatrix} \beta 11 & \beta 12 & \beta 13 & \beta 14 \\ 0 & \beta 22 & \beta 23 & \beta 24 \\ 0 & 0 & \beta 33 & \beta 34 \\ 0 & 0 & 0 & \beta 44 \end{bmatrix}$$

Tetapkan $\alpha_{ii} = 1$ untuk i = 1,....,N

Untuk j = 1,2,3,...,N lakukan dua prosedur yaitu :

Untuk i = 1,2,...,j gunakan rumus :

i < j $\quad \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \ldots + \alpha_{ii}\beta_{ij} = a_{ij}$

i = j $\quad \alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \ldots + \alpha_{ii}\beta_{ij} = a_{ij}$

$\alpha_{ii} = 1 \quad$ i = 1, ... , N

sehingga : $\beta_{ij} = \alpha_{ij} - \sum_{k=1}^{i-1} \alpha ik \beta kj$

Untuk i = j +1, j + 2, ... , N gunakan rumus i > j $\alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} + \ldots + \alpha_{ii}\beta_{ij} = a_{ij}$

sehingga :

$$\alpha_{ij} = \frac{1}{\beta jj}\left[a_{ij} - \sum_{k=1}^{j-1} \alpha ik \beta kj \right]$$

Lakukan langkah diatas untuk setiap j

Setelah diketahui harganya maka matrik disusun menjadi :

$$\begin{bmatrix} \beta 11 & \beta 12 & \beta 13 & \beta 14 \\ \alpha 21 & \beta 22 & \beta 23 & \beta 24 \\ \alpha 31 & \alpha 32 & \beta 33 & \beta 34 \\ \alpha 41 & \alpha 42 & \alpha 43 & \beta 44 \end{bmatrix}$$

Setelah dilakukan invers maka akan diperoleh matrik yang kemudian dikalikan dengan matrik fungsi defleksi.

Untuk memperoleh harga defleksi yang minimum, dilakukan pengaturan distribusi volume tangki ballast. Oleh karena itu digunakan metode optimasi dengan jumlah kuadrat defleksi sebagi fungsi obyektif.

2.3. METODE OPTIMASI

Metode Optimasi adalah suatu metode yang digunakan untuk mencari harga optimum dari suatu fungsi. Dalam menentukan titik optimum dari suatu fungsi terdapat banyak metode yang dapat digunakan.

Dalam Tugas Akhir ini digunakan metode Anealing. Pemilihan ini dikarenakan bila dalam suatu fungsi terdapat banyak lembah yang memuat titik minimum lokal maka metode ini dapat keluar dan akan mencari titik minimum global yang sebenarnya.

Berikut akan sedikit dibahas tentang beberapa metode optimasi dengan menggunakan variabel banyak.

2.3.1. Metode Downhill Simplex

Metode ini didasarkan pada Nelder dan Mead. Metode ini hanya mengevaluasi fungsi dan tidak perlu menurunkannya. Hal ini sangat tidak efisien bila dilihat dari jumlah fungsi yang harus dievaluasi. Namun demikian metode ini masih menjadi pilihan yang terbaik untuk menyelesaikan masalah komputasi dengan beban perhitungan yang ringan (lihat Press, 1989).

2.3.2. Metode Direction Set (Powell)

Metode ini tidak melibatkan perhitungan eksplisit dari gradien. Algoritma linmin didefinisikan sebagai berikut : input berupa vektor P dan vektor n serta fungsi f, cari skalar λ untuk meminimalkan $f(P + \lambda n)$. Ganti P dengan $P + \lambda n$. Ganti n dengan λn.

Ambil sekumpulan unit vektor e_1, e_2, e_3, ..., e_n sebagai set of direction. Dengan menggunakan *linmin*, fungsi digerakkan ke arah pertama menuju titik minimum, kemudian dari titik tersebut ke arah kedua menuju titik minimum kedua dan seterusnya.

Kelemahan dari cara diatas adalah bila fungsi yang akan diminimalkan merupakan fungsi dua dimensi dimana arah ke harga minimum merupakan lembah yang sempit sehingga untuk menuju ke harga minimum diperlukan banyak step. Hal ini tentu saja memperlama waktu optimasi.

Dalam metode Powell, cara yang digunakan yaitu dengan mengupdate atau memperbaharui set of direction bersamaan dengan berjalannya metode (lihat Press,1989).

2.3.3. Metode Conjugate Gradien

Metode ini digunakan jika harga fungsi f(P) dan gradien (vektor turunan parsial pertama) $\Delta f(P)$ dapat dihitung pada titik P dengan dimensi N tertentu.

Dua metode conjugate gradien yang umum adalah metode Fletcher-Reeves dan metode Polak-Ribiere. Kedua metode tersebut didasarkan pada teorema berikut :

Bila A adalah matrik n x n, definit positif dan simetris. Jika g_0 adalah vektor arbitrary dan $h_0 = g_0$. Untuk i = 0,1,2,...., tentukan dua vektor berikut :

$$g_{i+1} = g_i - \lambda_i A . h_i \qquad h_{i+1} = g_{i+1} + \gamma_i h_i$$

dimana λ_i dan γ_i dipilih sedemikian hingga $g_{i+1} . g_i = 0$ dan $h_{i+1} . h_i = 0$

sehingga :

$$\lambda_i = \frac{gi.gi}{gi.A.hi}$$

$$\gamma_i = -\frac{gi+1.A.hi}{hi.A.hi}$$

Pembahasan lebih lanjut dapat dilihat pada Polak.

2.3.4. Metode Metrik Variabel

Tujuan dari metode ini adalah mengakumulasikan informasi dari garis minimasi yang berurutan sehingga menuju ke bentuk kuadratik minimum dalam dimensi N.

Metrik variabel menggunakan pendekatan yang berbeda dari metode conjugate gradien yaitu dengan menyimpan dan mengupdate informasi yang dikumpulkan. Sehingga metode ini hanya memerlukan matrik N x N.

Ada dua metode metrik variabel yang sering digunakan yaitu algoritma Davidon - Fletcher - Powell dan algoritma Broyden - Fletcher - Goldfarb - Shanno. Perbedaan keduanya hanya terletak pada round-off error dan toleransi konvergensi (lihat Press, 1989).

2.3.5. Metode Anealing

Metode merupakan teknik yang sekarang banyak digunakan untuk menyelesaikan problem optimasi dengan skala besar.

Inti metode ini adalah analogi dengan termodinamika yaitu cara fluida atau cairan membeku dan mengkristal atau logam menjadi dingin dan aneal. Seperti kita ketahui, molekul fluida bergerak bebas satu terhadap yang lain. Bila cairan tersebut melepaskan panasnya secara perlahan-lahan, mobilitas thermal akan hilang. Atom-atom akan membentuk kristal murni. Kristal ini merupakan kondisi energi minimum untuk sistem tersebut. Yang mengagumkan adalah alam mampu menemukan kondisi minimum tersebut untuk sistem yang membeku secara perlahan-lahan. Tetapi proses pembekuan berlangsung secara drastis maka fluida tersebut tidak akan mencapai kondisi minimum tadi malah akan berakhir pada kondisi dengan tingkat energi yang lebih tinggi.

Jadi esensi dari proses ini adalah pembekuan perlahan-lahan yang akan memastikan tercapainya kondisi minimum.

Meski analogi tersebut tidak sempurna tetapi kalau kita melihat metode terdahulu seperti metode Powell, metode downhill simplex dll menunjukkan proses pembekuan yang berlangsung cepat. Mengapa demikian? Karena dalam metode-metode tersebut kita selalu berusaha untuk menuruni lembah sejauh mungkin. Hal ini mengakibatkan titik minimum yang diperoleh adalah titik minimum lokal bukan global. Algoritma optimasi alam didasarkan atas prosedur yang berbeda yang disebut distribusi probabilitas Boltzmann,

$$\text{Prob}(E) \sim \exp(-E/kT)$$

Persamaan diatas mengekspresikan ide bahwa sistem pada keseimbangan thermal pada suhu T mempunyai energinya sendiri yang secara probabilitas didistribusikan

pada semua tingkatan energi E. Walaupun pada suhu yang rendah terdapat kemungkinan meski sangat kecil bahwa sistem berada dalam tingkat energi tinggi. Oleh karena itu selalu terdapat kemungkinan bagi sistem untuk mencapai titik minimum global. K (konstanta Boltzmann) adalah konstanta yang menghubungkan temperatur dengan energi. Dengan kata lain sistem dapat naik dan juga turun.

Pembahasan lebih lengkap dapat dilihat pada (Press,1989).

BAB 3

PEMBUATAN PROGRAM

3.1. PROGRAM YANG DIGUNAKAN

Dalam Tugas Akhir ini program yang digunakan adalah bahasa Delphi versi 3. Bahasa program ini merupakan bahasa program tingkat tinggi karena bahasa program ini mudah untuk dipahami dan banyak digunakan. Dalam pembuatan sebuah program, Delphi menggunakan sistem yang disebut RAD (Rapid Application Development). Sistem ini memanfaatkan bahasa pemrograman visual yang membuat seorang programer lebih mudah mendesain tampilan program (user interface). Dengan menggunakan cara ini, merancang program yang bekerja dalam sistem operasi Windows lebih mudah. Dengan bahasa pemrograman non visual, programer harus merancang atau mendesain tampilan program dan mengatur bagaimana user berinteraksi dengan program tersebut seperti membuat efek penekanan tombol, mengatur pergerakan mouse dan sebagainya.

Dibandingkan dengan bahasa pemrograman visual lainnya seperti Visual Basic, Delphi mempunyai beberapa kelebihan. Karena Delphi dibangun dari bahasa Pascal yang merupakan bahasa pemrograman non visual paling banyak dipakai maka bagi para programer yang sebelum akrab dengan Pascal tidak akan menemui kesulitan untuk pindah ke Delphi.

Program dibagi dalam beberapa prosedur. Pembagian ini bertujuan bila terjadi penambahan atau perubahan rutin dapat dilakukan dengan hanya mengganti rutin yang ada didalam prosedur tersebut. Selain itu juga untuk mempermudah analisa alur program (Okianto, 1997).

Dalam Tugas Akhir ini, flow chart yang digunakan merupakan pengembangan dari flow chart yang terdapat pada Tugas Akhir Yani Rusihadi (Rusihadi, 1997).

3.2. FLOW CHART PROGRAM UTAMA

Jumlah dari keel blok ditentukan terlebih dahulu yaitu sebanyak 20 buah. Lalu harga-harga untuk panjang dok, lebar dok, lcg kapal dan lcg dok ditetapkan terlebih dahulu. Input data dibaca dari file teks yang berisikan data berat dok, volume tangki ballast, harga reaksi kapal, harga buoyancy dok, harga inersia kapal dan harga inersia dok. Data ini kemudian merupakan input dari prosedur Reaksi.

Jalannya proses adalah sebagai berikut : input data merupakan asumsi awal yang akan digunakan untuk menghitung.

Harga defleksi yang dihasilkan oleh prosedur REAKSI dicek oleh prosedur ANEAL untuk mengetahui apakah harga tersebut sudah memenuhi parameter yang ditentukan (dapat dikatakan optimum).

Apabila memenuhi maka langkah yang berhasil dicatat dalam variabel *Sukses* sehingga nantinya dapat diketahui jumlah langkah yang diperlukan untuk mencapai optimum.

Setelah itu dilakukan lagi pengaturan harga tangki ballast dalam prosedur ACAK. Dalam prosedur ini dilakukan pengacakan volume tiap tangki ballast dengan

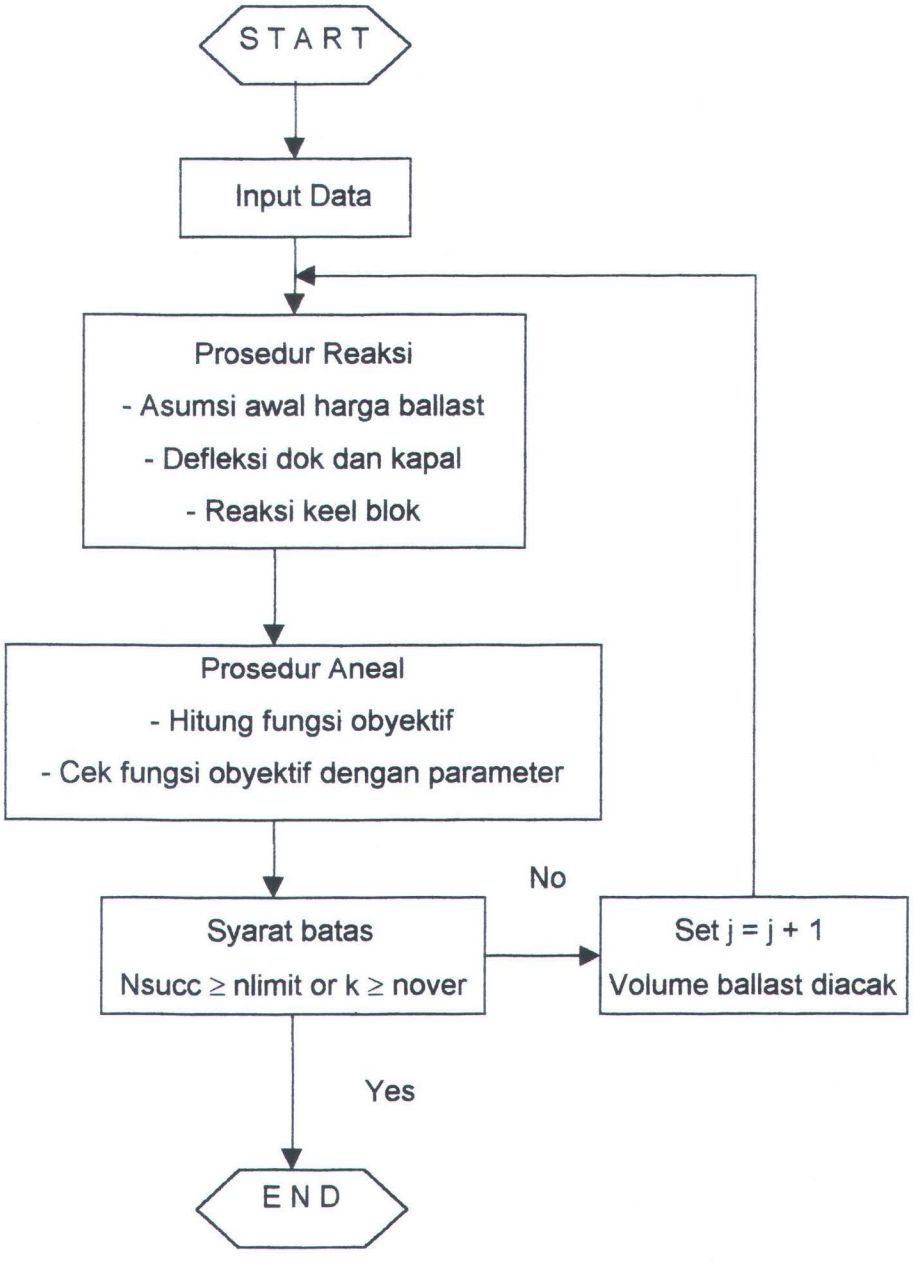
memanfaatkan perintah Random. Tentu saja terdapat batas maksimum dan minimum dari volume tersebut. Pembatasan dilakukan pada tinggi air ballast karena panjang dan lebar tangki ballast adalah konstan.

Apabila tidak memenuhi maka volume tangki ballast juga diacak.

Setelah volume tangki ballast diatur ulang, prosedur REAKSI dijalankan kembali.

Harga defleksi yang diperoleh dari prosedur tersebut diproses lagi dalam prosedur ANEAL untuk dibandingkan dengan harga defleksi sebelumnya.

Proses optimasi diatas diulang terus menerus hingga harga variabel k yang menyatakan banyaknya iterasi total lebih besar dari atau sama dengan harga variabel nover yang menyatakan jumlah maksimum iterasi total. Selain itu juga dibatasi hingga harga variabel nsucc yang menyatakan jumlah iterasi yang berhasil lebih besar dari atau sama dengan harga variabel nlimit yang menyatakan jumlah maksimum iterasi yang berhasil (lihat Lampiran, Prosedur ReaksiOptimum).



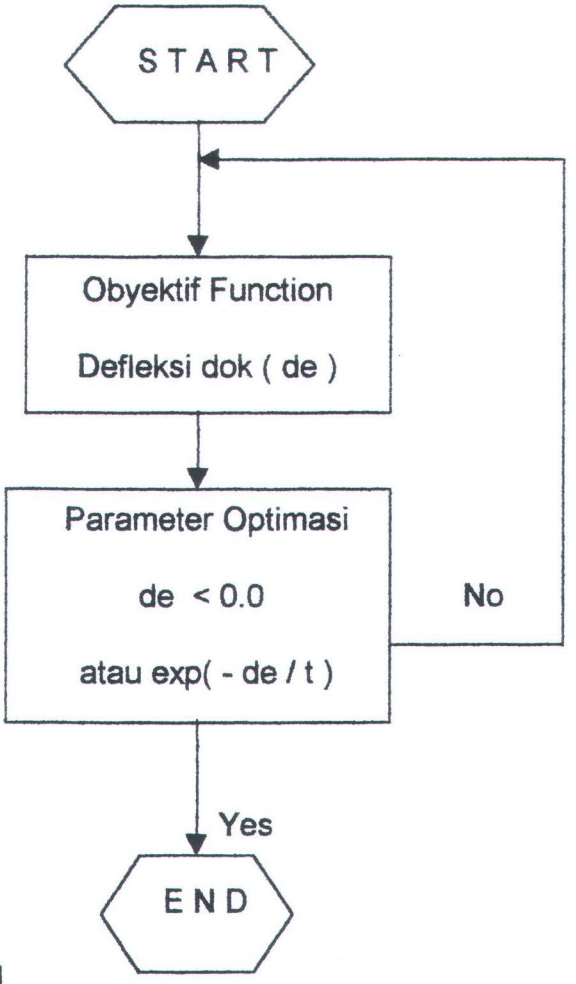
3.3. FLOW CHART METRO

Prosedur ini adalah prosedur untuk mengecek apakah harga yang dihasilkan memenuhi syarat optimum. Dikatakan optimum bila harga variabel ans berharga true.

Input dari prosedur ini adalah total kuadrat defleksi yang merupakan fungsi obyektif. Bila harga defleksi kurang dari nol maka variabel ans akan berharga true dan proses dilanjutkan lagi dengan mengacak volume tangki ballast yang nantinya merupakan input untuk prosedur reaksi selanjutnya.

Dan bila harga defleksi lebih besar dari nol maka variabel ans berharga true dengan probabilitas exp(defleksi / t) dimana t adalah suhu yang harganya telah ditentukan terlebih dahulu.

Output atau hasil berupa variabel ans yang bernilai true bila fungsi obyektif kurang dari nol atau dengan probabilitas exp(-de/t) dimana de adalah fungsi obyektif dan t adalah suhu yang harganya telah ditentukan terlebih dahulu (lihat Lampiran, prosedur Anealing).



3.4. FLOW CHART REAKSI

Prosedur ini digunakan untuk menghitung harga defleksi yang terjadi dengan volume tangki ballast sebagai variabel.

Untuk menghitung defleksi kapal dilakukan langkah-langkah berikut :

$$F(x)_{kapal} = \text{Reaksi keel blok koreksi } (x) - LWT_{kapal}(x)$$

Dari beban yang bekerja dapat dicari harga gaya lintang V(x), harga momen M(x), harga slope q(x) dan harga defleksi w(x).

$$V_{kapal\ i=x} = \sum_{i=0}^{x-1} (q_{kapal\ i,i+1} * 1)$$

dimana : $V_{kapal\ i=x}$ = harga gaya lintang kapal pada station ke-x

$q_{kapal\ i,i+1}$ = harga beban merata kapal pada station ke-i sampai i+1

l = jarak station

$$M_{kapal\ i=x} = \sum_{i=0}^{x} \left(\left(\frac{V_{kapali} + V_{kapali+1}}{2} \right) * 1 \right)$$

dimana : $M_{kapal\ i=x}$ = harga momen lengkung kapal pada station ke-x

$V_{kapal\ i}$ = harga gaya lintang kapal pada station ke-i

$V_{kapal\ i+1}$ = harga gaya lintang kapal pada station ke-i+1

$$\theta_{kapal\ i=x} = \sum_{i=0}^{x} \frac{1}{EI_{kapali}} \left(\left(\frac{M_{kapali} + M_{kapali+1}}{2} \right) * 1 \right)$$

dimana : $\theta_{kapal\ i=x}$ = harga slope kapal pada station ke-x

$M_{kapal\ i}$ = harga momen lengkung kapal pada station ke-i

$M_{kapal\ i+1}$ = harga momen lengkung kapal pada station ke-i+1

E = modulus elastisitas baja (200×10^9 N/m^2)

$I_{kapal\ i}$ = inersia kapal pada station ke-i (m^4)

$$w_{kapal\ i=x} = \sum_{i=0}^{x} \left(\left(\frac{\theta_{kapali} + \theta_{kapali+1}}{2} \right) * 1 \right)$$

dimana : $w_{kapal\ i=x}$ = harga defleksi kapal pada station ke-x

$\theta_{kapal\ i}$ = harga slope kapal pada station ke-i

$\theta_{kapal\ i+1}$ = harga slope kapal pada station ke-i+1

Sedang untuk memperoleh defleksi floating dock dilakukan langkah berikut :

$F(x)_{dok}$ = koreksi buoyancy(x) – reaksi blok(x) – LWT_{dok}(x) – ballast(x)

$$V_{dok\ i=x} = \sum_{i=0}^{x-1} (q_{dok\ i,i+1} * 1)$$

dimana : $V_{dok\ i=x}$ = harga gaya lintang dok pada station ke-x

$q_{dok\ i,i+1}$ = harga beban merata dok pada station ke-i sampai i+1

l = jarak station

$$M_{dok\ i=x} = \sum_{i=0}^{x} \left(\left(\frac{V_{doki} + V_{doki+1}}{2} \right) * 1 \right)$$

dimana : $M_{dok\ i=x}$ = harga momen lengkung dok pada station ke-x

$V_{dok\ i}$ = harga gaya lintang dok pada station ke-i

$V_{dok\ i+1}$ = harga gaya lintang dok pada station ke-i+1

$$\theta_{dok\ i=x} = \sum_{i=0}^{x} \frac{1}{EI_{doki}} \left(\left(\frac{M_{doki} + M_{doki+1}}{2} \right) * 1 \right)$$

dimana : $\theta_{dok\ i=x}$ = harga slope dok pada station ke-x

$M_{dok\ i}$ = harga momen lengkung dok pada station ke-i

$M_{dok\ i+1}$ = harga momen lengkung dok pada station ke-i+1

E = modulus elastisitas baja (200×10^9 N/m^2)

$I_{dok\ i}$ = inersia dok pada station ke-i (m^4)

$$W_{dok\ i=x} = \sum_{i=0}^{x} \left(\left(\frac{\theta_{doki} + \theta_{doki+1}}{2} \right) * 1 \right)$$

dimana : $w_{dok\ i=x}$ = harga defleksi dok pada station ke-x

$\theta_{dok\ i}$ = harga slope dok pada station ke-i

$\theta_{dok\ i+1}$ = harga slope dok pada station ke-i+1

Dari kedua harga defleksi kapal dan dok, didapatkan selisih defleksi w(x).

$$w(x) = w_{dok}(x) - w_{kapal}(x)$$

Kemudian dapat diperoleh matrik kolom defleksi { w(x) }. Untuk menyelesaikan persamaan dengan variabel banyak digunakan metode Newton Raphson.

Persamaan Newton Raphson :

$$\begin{bmatrix} \frac{dw_1(x_1)}{dx_1} & \frac{dw_1(x_2)}{dx_2} & \cdots & \frac{dw1(x_k)}{dx_k} \\ \frac{dw_2(x_1)}{dx_1} & \frac{dw_2(x_2)}{dx_2} & \cdots & \frac{dw2(x_k)}{dx_k} \\ \frac{dw_3(x_1)}{dx_1} & \frac{dw_3(x_2)}{dx_2} & \cdots & \frac{dw3(x_k)}{dx_k} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{dwi(x_1)}{dx_1} & \frac{dwi(x_2)}{dx_2} & \cdots & \frac{dwi(x_k)}{dx_k} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdots \\ h_k \end{bmatrix} = - \begin{bmatrix} w_1(x_i) \\ w_2(x_i) \\ \cdots \\ w_n(x_i) \end{bmatrix}$$

$\frac{dw_1(x_4)}{dx_4}$ adalah $\frac{w_1(x_4 + \Delta x_4) - w_1(x_4)}{\Delta x_4}$

dimana :

$w_1(x_4)$ adalah selisih defleksi dok dan defleksi kapal pada station 1

$w_1(x_4 + \Delta x_4)$ adalah selisih defleksi dok dan kapal pada station 1 dengan harga reaksi keel blok pada station 4 ditambah Δx_4 sedang harga reaksi keel blok lainnya tetap yaitu harga reaksi keel blok yang sudah dikoreksi.

Karena $w_1 = w_{dok\,1} - w_{kapal\,1}$ maka :

$$\frac{dw_1(x_4)}{dx_4} = \frac{W_{dok1}(x_4 + \Delta x_4) - W_{dok1}(x_4)}{\Delta x_4} - \frac{W_{kapal1}(x_4 + \Delta x_4) - W_{kapal1}(x_4)}{\Delta x_4}$$

dimana :

$w_{dok\,1}\,(x_4+\Delta x_4)$ adalah defleksi dok pada station 1 bila harga reaksi keel blok pada station 4 ditambah Δx_4 sedang harga reaksi keel blok lainnya tetap.

$w_{dok\,1}\,(x_4)$ adalah defleksi dok pada station 1.

$w_{kapal\ 1}(x_4+\Delta x_4)$ adalah defleksi kapal pada station 1 bila harga reaksi keel blok pada station 4 ditambah Δx_4 sedang harga reaksi keel blok lainnya tetap.

$w_{kapal\ 1}(x_4)$ adalah defleksi kapal pada station 1.

Untuk mendapatkan matrik Jacobian :

Reaksi keel blok pada station 1 ditambah dengan Δx_1 sedang harga reaksi keel blok lainnya tetap. Operasi ini dilaksanakan pada kapal dan dok. Selanjutnya dihitung harga defleksi akibat perubahan tersebut dari station 1 sampai station akhir.

$w_{dok\ 1}(x_1 + \Delta x_1)$ ..... $w_{dok\ 100}(x_1 + \Delta x_1)$

$w_{kapal\ 1}(x_1 + \Delta x_1)$ ..... $w_{kapal\ 100}(x_1 + \Delta x_1)$

Kemudian harga defleksi tersebut dikurangi dengan harga defleksi tanpa penambahan reaksi keel blok.

$w_{dok\ 1}(x_1 + \Delta x_1) - w_{dok\ 1}(x_1)$ ..... $w_{dok\ 100}(x_1 + \Delta x_1) - w_{dok\ 100}(x_1)$

$w_{kapal\ 1}(x_1 + \Delta x_1) - w_{kapal\ 1}(x_1)$ ..... $w_{kapal\ 100}(x_1 + \Delta x_1) - w_{kapal\ 100}(x_1)$

Lalu harga defleksi dok dikurangi harga defleksi kapal.

$$\frac{dw_1(x_1)}{dx_1} = \frac{w_{dok1}(x_1 + \Delta x_1) - w_{dok1}(x_1)}{\Delta x_1} - \frac{w_{kapal1}(x_1 + \Delta x_1) - w_{kapal1}(x_1)}{\Delta x_1}$$

Langkah tersebut diteruskan sampai station ke 100 hingga didapatkan harga satu kolom matrik Jacobian. Untuk memperoleh matrik Jacobian, dilakukan perhitungan perubahan reaksi keel blok sampai station ke 100.

Dari matrik Jacobian kemudian diinvers sehingga diperoleh matrik Jacobian invers.

$$[J] \Rightarrow [J]^{-1}$$

Harga reaksi keel blok baru didapatkan dari pengurangan reaksi keel blok lama oleh hasil perkalian matrik jacobian invers dengan matrik kolom selisih defleksi.

$$\{X_{k+1}\} = \{X_k\} - [J]^{-1} * \{w\}$$

Pada iterasi pertama harga $\{X_k\}$ adalah harga reaksi keel blok awal yang telah dikoreksi dan selanjutnya harga reaksi keel blok baru menjadi reaksi keel blok lama.

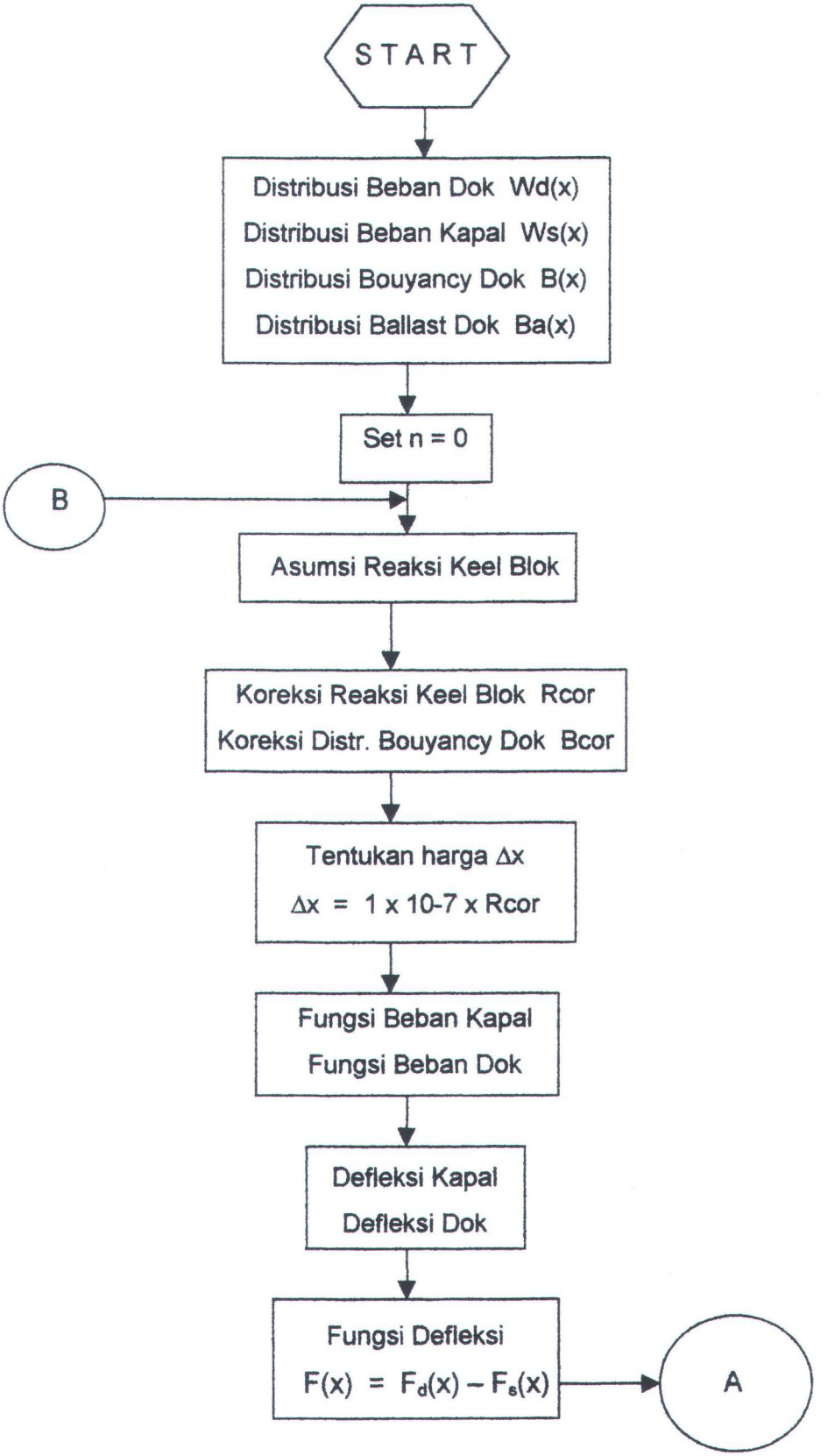
$$\{X_{k+1}\} = \{X_k\}$$

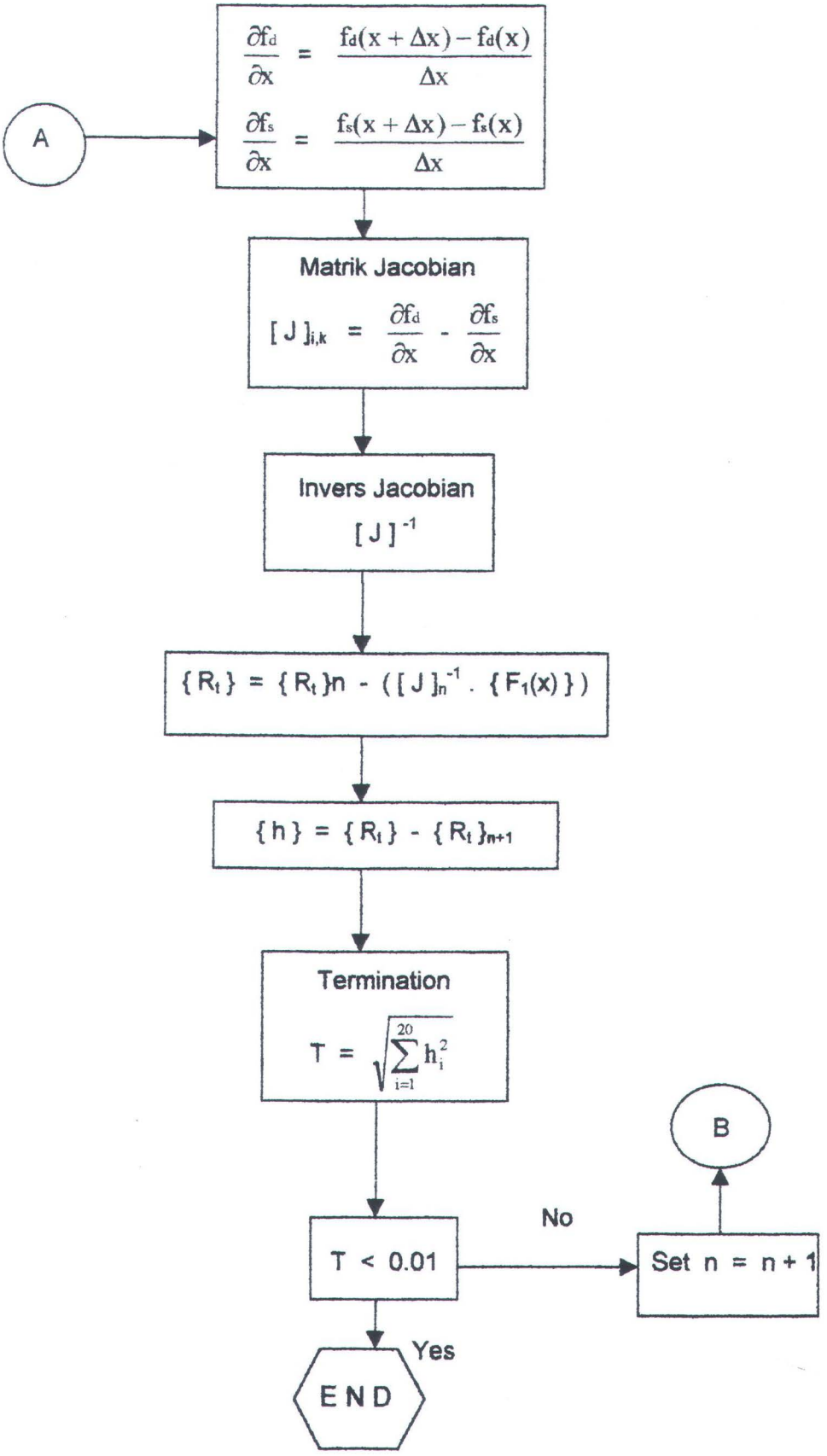
Sebagai parameter bahwa iterasi dapat dihentikan adalah defleksi kapal dan defleksi dok sama atau mendekati harga yang ditentukan (T).

$$T = \sqrt{\sum_{k=1}^{20}(h_k^2)} \leq 0.01$$

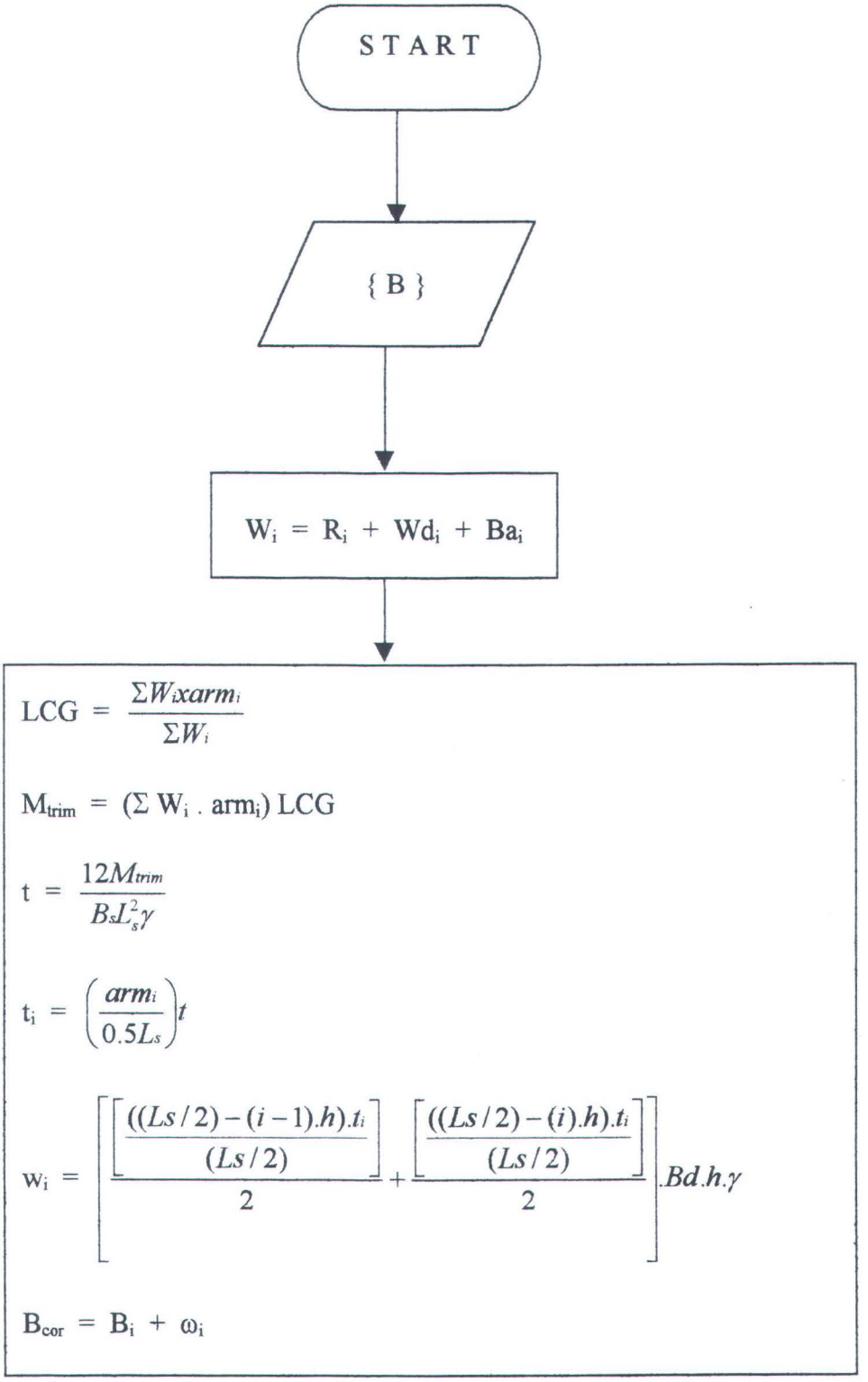
Dimana : $h_k = \{X_{k+1}\} - \{X_k\}$

Lihat lampiran, prosedur ReaksiOptimum.

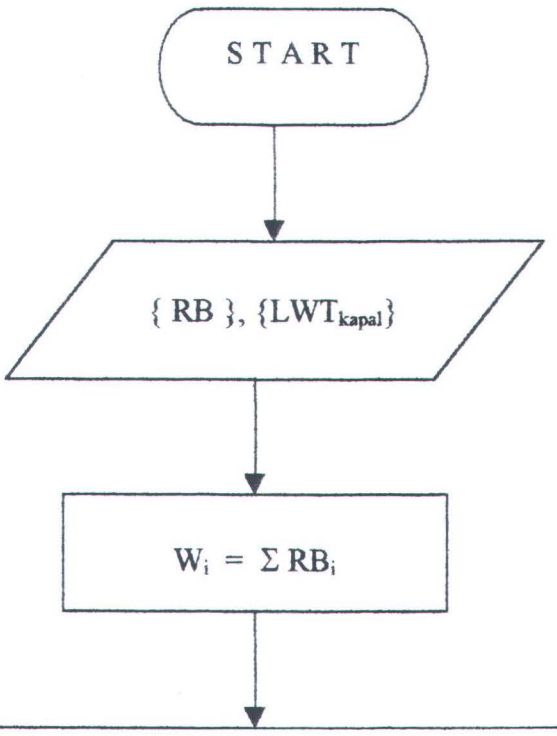

START
Distribusi Beban Dok Wd(x)
Distribusi Beban Kapal Ws(x)
Distribusi Bouyancy Dok B(x)
Distribusi Ballast Dok Ba(x)
Set n = 0
B
Asumsi Reaksi Keel Blok
Koreksi Reaksi Keel Blok Rcor
Koreksi Distr. Bouyancy Dok Bcor
Tentukan harga Δx
Δx = 1 x 10-7 x Rcor
Fungsi Beban Kapal
Fungsi Beban Dok
Defleksi Kapal
Defleksi Dok
Fungsi Defleksi
F(x) = Fd(x) – Fs(x)
A


A
$\frac{\partial f_d}{\partial x} = \frac{f_d(x+\Delta x) - f_d(x)}{\Delta x}$
$\frac{\partial f_s}{\partial x} = \frac{f_s(x+\Delta x) - f_s(x)}{\Delta x}$
Matrik Jacobian
$[J]_{i,k} = \frac{\partial f_d}{\partial x} - \frac{\partial f_s}{\partial x}$
Invers Jacobian
$[J]^{-1}$
$\{R_t\} = \{R_t\}n - ([J]_n^{-1} . \{F_1(x)\})$
$\{h\} = \{R_t\} - \{R_t\}_{n+1}$
Termination
$T = \sqrt{\sum_{i=1}^{20} h_i^2}$
T < 0.01
No
Set n = n + 1
B
Yes
END

3.5. FLOW CHART KOREKSI BUOYANCY DOCK



3.6. FLOWCHART KOREKSI REAKSI KEEL BLOCK



$$LCG = \frac{\Sigma LWTixarm_i}{\Sigma LWT_i}$$

$$M_{trim} = (\Sigma\ LWT_i)\ LCG$$

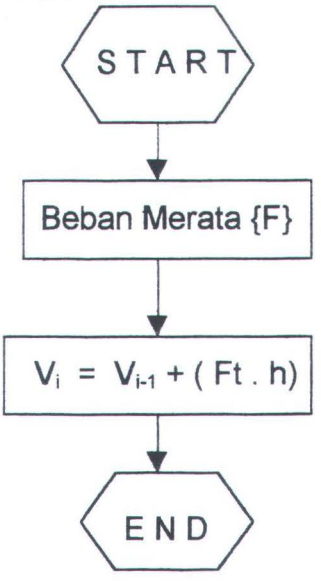
$$q_b = \left(\frac{6 * M_{trim}}{L_S^2}\right) + \left(\frac{W_t}{L_s}\right)$$

$$q_d = \left(\frac{6 * M_{trim}}{L_S^2}\right)$$

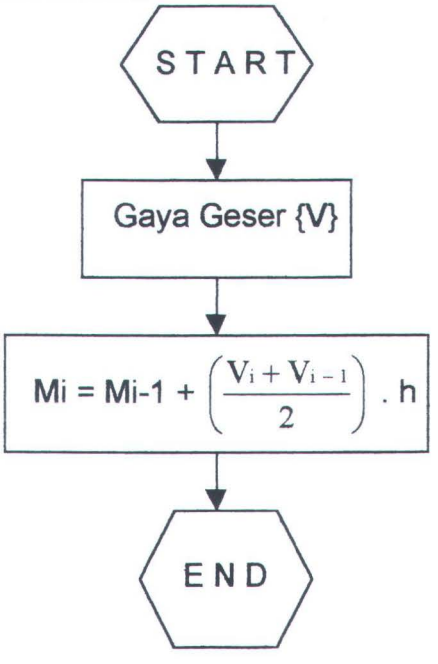
$$T_i = \left[\frac{L_s - (i-1) * h}{L_s}\right] * (q_b - q_d) + q_d$$

$$RB_{cor} = \left[\frac{T_i + T_{i+1}}{2}\right] * h$$

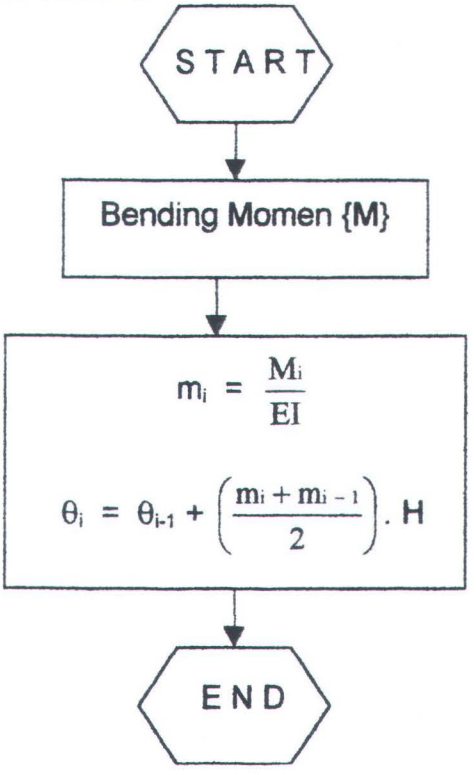
3.7. FLOWCHART SHEARING FORCE



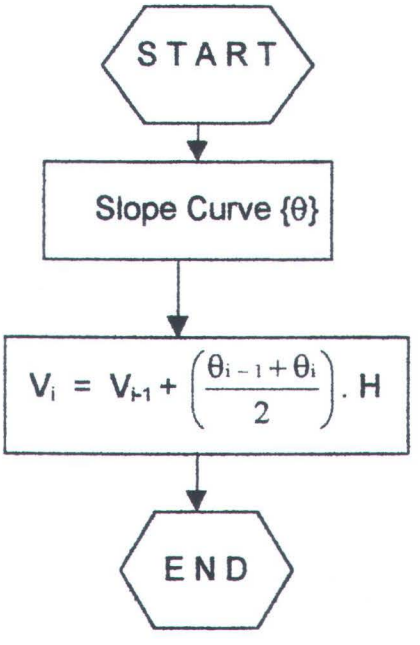
3.8. FLOWCHART BENDING MOMENT



3.9. FLOWCHART SLOPE CURVE



3.10. FLOWCHART DEFLECTION CURVE



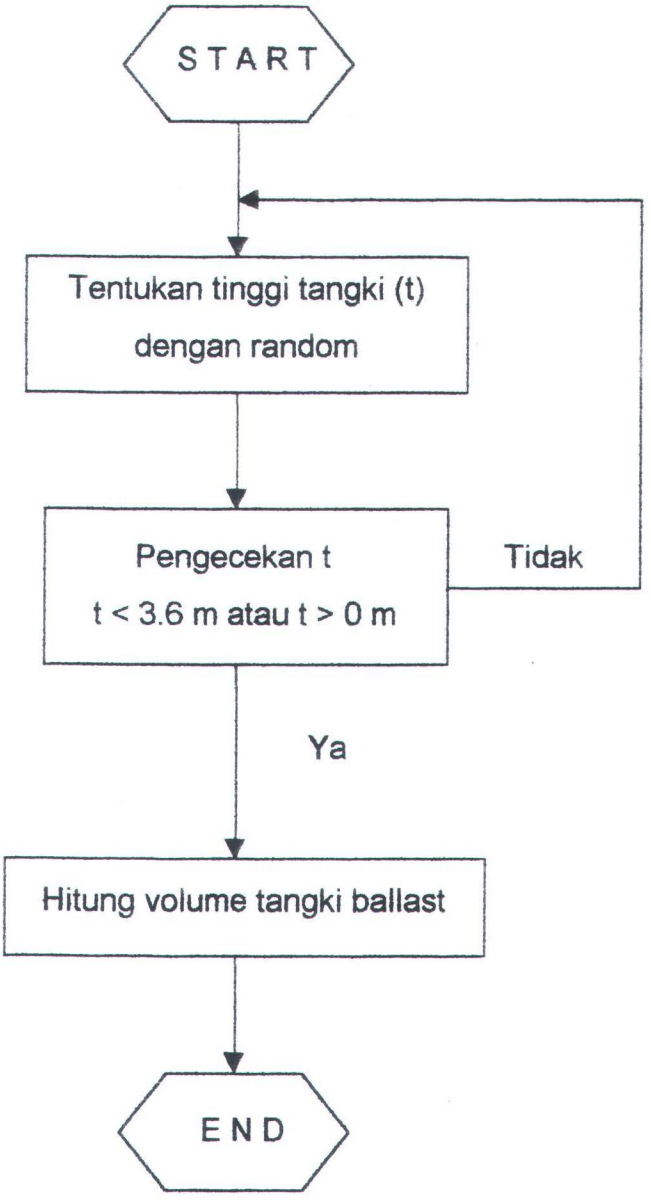
3.11. PROSEDUR ACAK

Input untuk prosedur ini adalah tinggi tangki ballast yang dipanggil melalui perintah Random.

Prosedur ini digunakan untuk memperoleh harga volume tiap tangki ballast yang berbeda antara tangki satu dengan yang lain.

Harga lebar dan panjang tiap tangki adalah konstan sehingga yang menjadi variabel adalah tinggi air di tiap tangki. Agar sesuai dengan kondisi sebenarnya tinggi air diberi batasan minimal dan maksimal. Batasan minimal adalah 0 m sedangkan tinggi maksimal adalah 3.6 m.

Untuk memperoleh harga yang berbeda-beda digunakan perintah Random. Supaya harga yang diperoleh tidak sama sebelum perintah Random dieksekusi maka dijalankan dahulu perintah Randomize. Perintah ini bertujuan mengambil bibit (seed) yang berbeda-beda sehingga nantinya hasil akhir dari Random tidak sama (lihat lampiran, prosedur ReaksiOptimum).


START
Tentukan tinggi tangki (t)
dengan random
Pengecekan t
t < 3.6 m atau t > 0 m
Tidak
Ya
Hitung volume tangki ballast
END

BAB 4

HASIL PROGRAM

4.1. TAMPILAN PROGRAM

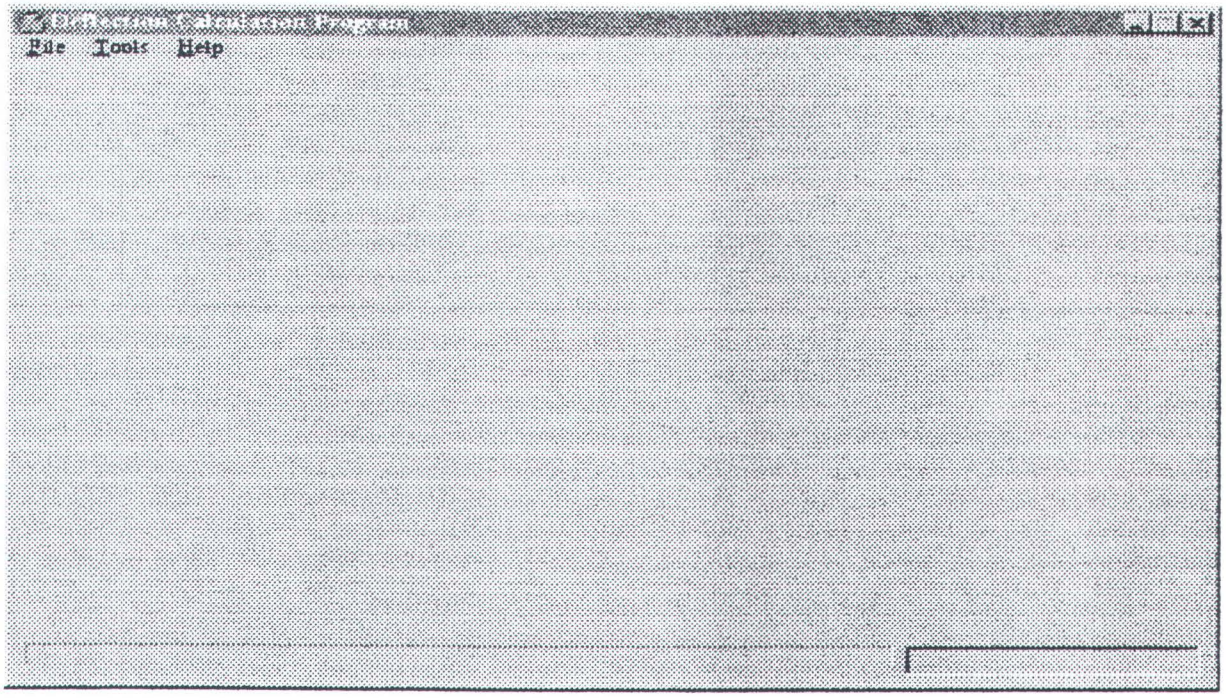
Program dibuat dalam bahasa Delphi versi 3.0. Setelah dieksekusi, program akan menampilkan menu utama. Menu ini terdiri dari File, Tool dan Help.

Menu File mempunyai beberapa sub menu yaitu sub menu Open untuk membuka file data yang berupa database, sub menu Close untuk menutup file data yang sedang dibuka dan sub menu Exit yang berfungsi untuk keluar dari program dan kembali ke Windows.

Menu Tool terdiri dari sub menu Optimize dan sub menu Show Graphic. Sub menu Optimize digunakan untuk menjalankan proses optimasi dengan menggunakan file data yang sedang aktif. Sedang sub menu Show Graphic digunakan untuk menampilkan hasil optimasi dalam bentuk grafik.

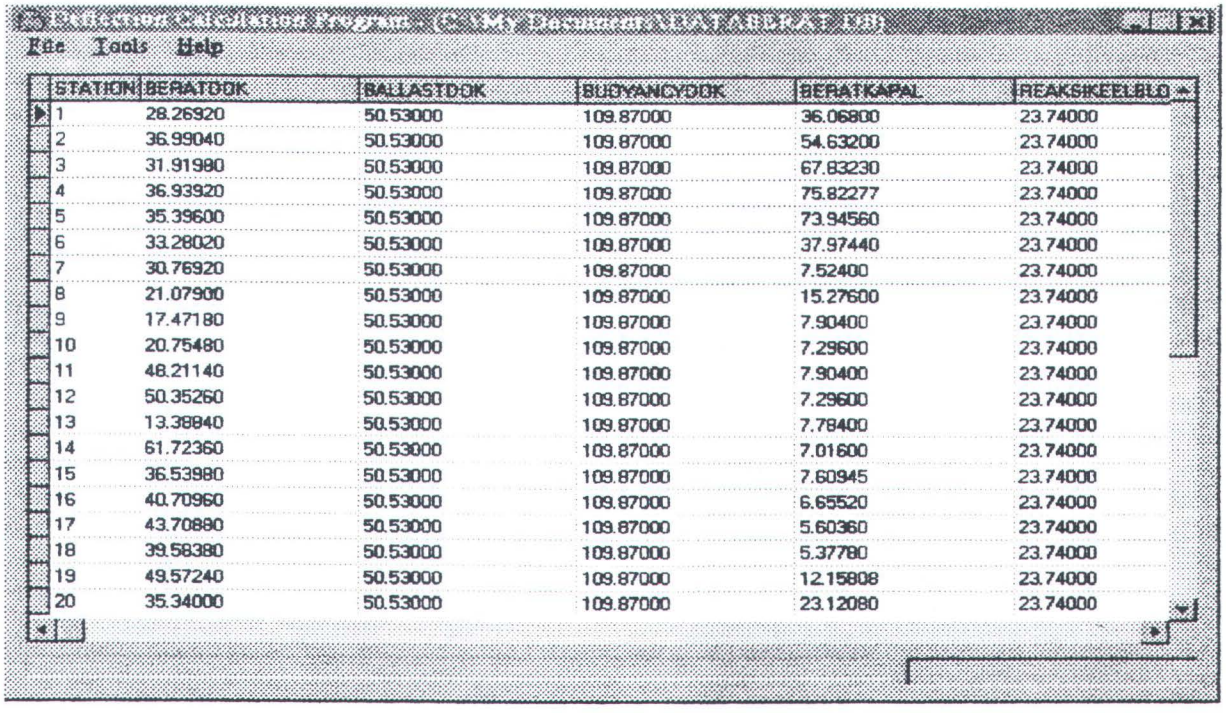
Menu Help digunakan untuk menampilkan keterangan mengenai program.

Berikut adalah tampilan menu utama dari program.



Gambar 4.1. Tampilan utama program

Berikut adalah tampilan setelah data beban dibuka :



STATION	BERATDOK	BALLASTDOK	BUOYANCYDOK	BERATKAPAL	REAKSIKEELBLG
1	28.26920	50.53000	109.87000	36.06800	23.74000
2	36.99040	50.53000	109.87000	54.63200	23.74000
3	31.91980	50.53000	109.87000	67.83230	23.74000
4	36.93920	50.53000	109.87000	75.82277	23.74000
5	35.39600	50.53000	109.87000	73.94560	23.74000
6	33.28020	50.53000	109.87000	37.97440	23.74000
7	30.76920	50.53000	109.87000	7.52400	23.74000
8	21.07900	50.53000	109.87000	15.27600	23.74000
9	17.47180	50.53000	109.87000	7.90400	23.74000
10	20.75480	50.53000	109.87000	7.29600	23.74000
11	48.21140	50.53000	109.87000	7.90400	23.74000
12	50.35260	50.53000	109.87000	7.29600	23.74000
13	13.38840	50.53000	109.87000	7.78400	23.74000
14	61.72360	50.53000	109.87000	7.01600	23.74000
15	36.53980	50.53000	109.87000	7.60945	23.74000
16	40.70960	50.53000	109.87000	6.65520	23.74000
17	43.70880	50.53000	109.87000	5.60360	23.74000
18	39.58380	50.53000	109.87000	5.37780	23.74000
19	49.57240	50.53000	109.87000	12.15808	23.74000
20	35.34000	50.53000	109.87000	23.12080	23.74000

Gambar 4.2. Tampilan data beban

Dari kedua grafik defleksi diatas ternyata diperoleh hasil :

Defleksi sebelum dioptimasi adalah 0.006775263 m

Defleksi setelah dioptimasi adalah $8.48 * 10^{-6}$ m

Titik maksimum defleksi setelah dioptimasi bergeser ke kanan dikarenakan adanya perbedaan distribusi ballast dengan kondisi awal. Perbedaan ini disebabkan adanya pengacakan tinggi ballast dalam tangki untuk memperoleh defleksi yang optimum.

Program ini dapat dikembangkan dengan ukuran utama kapal sebagai input dimana perubahan yang harus dilakukan adalah dalam hal penyebaran berat kapal dan model pembebanan pada dok. Kemudian jumlah keel blok dapat diperbanyak sehingga mendekati kondisi yang sebenarnya yaitu sebanyak 120 buah keel blok.

PROGRAM UTAMA

```
unit Utama;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, Grids;

type
  TFormTA = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Reaksi1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    Keluar1: TMenuItem;
    Optimasi1: TMenuItem;
    Hasil1: TMenuItem;
    procedure Keluar1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure Optimasi1Click(Sender: TObject);
    procedure Hasil1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormTA: TFormTA;

implementation

uses Pribadi,ReaksiOptimum,TampilHasil;

{$R *.DFM}

procedure TFormTA.Keluar1Click(Sender: TObject);
begin
    close;
end;

procedure TFormTA.About1Click(Sender: TObject);
begin
    FormTentang := TFormTentang.Create(Self);
    FormTentang.ShowModal;
```

```
    FormTentang.Free;
end;

procedure TFormTA.Optimasi1Click(Sender: TObject);
begin
    FormOptimasi := TFormOptimasi.Create(Self);
    FormOptimasi.ShowModal;
    FormOptimasi.Free;
end;

procedure TFormTA.Hasil1Click(Sender: TObject);
begin
    FormHasil := TFormHasil.Create(Self);
    FormHasil.ShowModal;
    FormHasil.Free;
end;

end.
```

PROSEDUR 2

```
unit prosedur2_cad;

interface
const
    max = 20;
    arm : array [1..20] of real =
    (47.5,42.5,37.5,32.5,27.5,22.5,17.5,12.5,7.5,2.5,-2.5,
    -7.5,-12.5,-17.5,-22.5,-27.5,-32.5,-37.5,-42.5,-47.5);
    stat : array [1..20] of byte = (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
    16,17,18,19,20);
type
   data   = array [1..max] of extended;
   integ  = array [1..max] of integer;
   matrix = array [1..max,1..max] of extended;

procedure stopper(nb      : integer;
                  vdel    : data;
                  var test : double);
procedure matr_mul(row_x, col_x, col_y : integer;
                   mx, my            : matrix;
                   var mz            : matrix);
procedure matr_sub(nb                : integer;
                   mat1, mat2        : matrix;
                   var mat3          : matrix);

implementation

procedure stopper;
var
  nn    : integer;
  summ  : double;
begin
    summ := 0;
    for nn := 1 to nb do
       summ := summ + sqr(vdel[nn]);
    test := sqrt(summ);
end;

procedure matr_mul;
var
  k,m,n  : integer;
begin
    for m := 1 to row_x do
    for n := 1 to col_y do
    begin
        mz[m,n] := 0;
```

```
        for k := 1 to col_x do
        mz[m,n] := mz[m,n]+(mx[m,k]*my[k,n]);
    end;
end;

procedure matr_sub;
var
  k, b   : integer;
begin
    for k:= 1 to nb do
    for b:= 1 to nb do
    mat3[k,b] := mat1[k,b]-mat2[k,b];
end;

end.
```

PROSEDUR 3

```
unit prosedur3_cad;

interface

uses prosedur2_cad;

procedure loading(nb                : integer;
          wd,bad,cor_bod,wedge : data;
          var load_dat       : data);
procedure loading1(nb               : integer;
          ilcg,spx,ls     : real;
          shipw           : data;
          rb              : data;
          var wedge       : data);
procedure loading2(nb               : integer;
          wd,bad,bod,wedge : data;
          dlcg,spx,br,beam : real;
          var cor_bod     : data);
procedure loading3(nb          : integer;
          shipw       : data;
          wedge       : data;
          var load_dat : data);
procedure shearing(nb          : integer;
          spx         : real;
          load_dat    : data;
          var shear_dat : data);
procedure bending(nb          : integer;
          spx         : real;
          shear_dat   : data;
          var bend_dat : data);
procedure sloping(nb          : integer;
          spx         : real;
          bend_dat    : data;
          ine         : data;
          young       : real;
          var slope_dat : data);
procedure curving(nb          : integer;
          spx         : real;
          slope_dat   : data;
          var cfd_dat : data);
procedure dock_cur(nb               : integer;
          wd,bad,bod,wedge  : data;
          dlcg,spx,br,beam  : real;
          ine1              : data;
          young1            : real;
          var cfs           : data;
```

```
                var cfd          : data);
procedure dock_cur2(nb                  : integer;
                wd,bad,cor_bod,wedge : data;
                spx                : real;
                ine1               : data;
                young1               : real;
                var cfs            : data;
                var cfd            : data);
procedure ship_cur(nb                : integer;
                ilcg,spx,ls      : real;
                shipw            : data;
                rb,wedge           : data;
                ine2             : data;
                young2             : real;
                var cfs          : data;
                var cfd          : data);
procedure ship_cur2(nb               : integer;
                shipw          : data;
                wedge            : data;
                spx            : real;
                ine2           : data;
                young2           : real;
                var cfs        : data;
                var cfd        : data);
procedure make_mat1(nb                  : integer;
                wd,bad,cor_bod,wedge : data;
                spx                : real;
                ine1               : data;
                young1               : real;
                delx               : data;
                var mat1             : matrix);
procedure make_mat2(nb                  : integer;
                spx                : real;
                ine2               : data;
                young2               : real;
                shipw              : data;
                wedge,delx           : data;
                var mat2           : matrix);
procedure mat_maker1(nb         : integer;
                dat1       : data;
                var mat_x1 : matrix);
procedure mat_maker2(nb         : integer;
                dat2       : data;
                var mat_x2 : matrix);
procedure jacobian(nb              : integer;
                mat1,mat2      : matrix;
                dat1,dat2,delx : data;
                var jac        : matrix);
```

```
implementation

procedure loading;
var
  p : integer;
begin
    for p := 1 to nb do
    begin
        load_dat[p] := (cor_bod[p]-(wedge[p]+wd[p]+bad[p]));
    end;
end;

procedure loading1;
type
   ship_data = record
           berat,mmt,qd,qb,lcg,lwt,brt,lwg : real;
           end;
var
  p,ns                    : integer;
  sdt                     : ^ship_data;
  load_tot,sum_tot,
  ttk_brt,jumlah_lwt,ttk_lwt,
  momen_lwt,jumlah_abo    : real;
  axx                     : real;
  beda                    : real;
  tinggi,abo,abo1,abo2,momen,
  ttbrt,lng,dist,dist1,dist2   : data;
begin
    new(sdt);
    sdt^.lwt := 0.00;
    for p := 1 to nb do
    begin
        sdt^.lwt := sdt^.lwt + shipw[p];
    end;
    sdt^.brt := 0.00;
    for p := 1 to nb do
    begin
        sdt^.brt := sdt^.brt + (shipw[p]*arm[p]);
    end;
    sdt^.lwg := sdt^.brt/sdt^.lwt;
    sdt^.berat := 0.00;
    for p := 1 to nb do
    begin
        sdt^.berat := sdt^.berat+rb[p];
    end;
    sdt^.mmt := 0.00;
    for p := 1 to nb do
    begin
        sdt^.mmt := sdt^.mmt+(rb[p]*arm[p]);
```

```
end;
sdt^.lcg := sdt^.mmt/sdt^.berat;
beda := sdt^.lwg-sdt^.lcg;
if (abs(beda) > 0.001) then
begin
    spx := ls/nb;
    ns := nb+1;
    axx := ((6*sdt^.berat)*(sdt^.lwg))/sqr(ls);
    sdt^.qd := ((sdt^.berat)/ls)-axx;
    sdt^.qb := ((sdt^.berat)/ls)+axx;
    for p := 1 to ns do
    tinggi[p] := ((ls-(p-1)*spx)/ls)*(sdt^.qb-sdt^.qd)+sdt^.qd;
    for p := 1 to nb do
    begin
        wedge[p] := ((tinggi[p]+tinggi[p+1])/2)*spx;
    end;
    load_tot := 0.00;
    for p := 1 to nb do
    load_tot := load_tot + wedge[p];
    for p := 1 to nb do
    abo1[p] := tinggi[p+1]*spx;
    for p := 1 to nb do
    abo2[p] := ((tinggi[p]-tinggi[p+1])*spx)/2;
    for p := 1 to nb do
    abo[p] := abo1[p]+abo2[p];
    for p := 1 to nb do
    momen[p] := abo2[p]*(spx/6);
    for p := 1 to nb do
    ttbrt[p] := momen[p]/(abo[p]);
    for p := 1 to nb do
    begin
        lng[p] := arm[p]+ttbrt[p];
    end;
    jumlah_abo := 0.00;
    for p := 1 to nb do
    begin
        jumlah_abo := jumlah_abo+abo[p];
    end;
    sum_tot := 0.00;
    for p := 1 to nb do
    begin
        sum_tot := sum_tot+(abo[p]*lng[p]);
    end;
    ttk_brt := sum_tot/jumlah_abo;
    for p := 1 to nb-1 do begin
       dist1[p] := wedge[p]*(0.5+((ttbrt[p]+(spx/2))/spx));
       dist2[p] := wedge[p]*(0.5-((ttbrt[p]+(spx/2))/spx));
       dist1[nb] := wedge[nb]*(0.5+((spx/2)-ttbrt[nb])/spx);
       dist2[nb] := wedge[nb]*(0.5-((spx/2)-ttbrt[nb])/spx);
```

```
        end;
        dist[1] := dist1[1];
        for p := 2 to nb-2 do
        begin
           dist[p] := dist2[p-1]+dist1[p];
           dist[nb-1] := dist2[nb-2]+dist1[nb-1]+dist2[nb];
           dist[nb] := dist1[nb]+dist2[nb-1];
        end;
        jumlah_lwt := 0.00;
        for p := 1 to nb do
        begin
           jumlah_lwt := jumlah_lwt+dist[p];
        end;
        momen_lwt := 0.00;
        for p := 1 to nb do
        begin
           momen_lwt := momen_lwt+(dist[p]*arm[p]);
        end;
        ttk_lwt := momen_lwt/jumlah_lwt;
        for p := 1 to nb do wedge[p] := dist[p];
    end else
    begin
        for p := 1 to nb do
        begin
           wedge[p] := rb[p];
        end;
    end;
    dispose(sdt);
end;

procedure loading2(nb              : integer;
                   wd,bad,bod,wedge : data;
                   dlcg,spx,br,beam : real;
                   var cor_bod      : data);
var
  p,ns                             : integer;
  trim_x,wedge1,abot1,abot2,
  abot,momen,ttbrt,lng,distr,distr1,distr2  : data;
  lcg,lcb                          : real;
  t_dt, p_dt                       : data;
  t_wx,summ,b_tx,m_trim,d_trim,b_dt,
  jumlah_baji,jumlah_abot,jumlah_wedge1,
  bouyancy,bou,momen_baji          : real;
begin
    for p := 1 to nb do
    p_dt[p] := wedge[p]+wd[p]+bad[p];
    t_wx := 0.00;
    for p := 1 to nb do t_wx := t_wx+p_dt[p];
    for p := 1 to nb do t_dt[p] := p_dt[p]*arm[p];
```

```
summ := 0.00;
for p := 1 to nb do summ := summ+t_dt[p];
lcg := summ/t_wx;
m_trim := t_wx*lcg;
d_trim := (6*m_trim)/(1.025*br*sqr(beam));
ns := nb+1;
spx := beam/nb;
for p := 1 to ns do
trim_x[p] := (((beam*0.5)-(p-1)*spx)/(beam*0.5))*d_trim;
for p := 1 to nb do
begin
   wedge1[p] := ((trim_x[p]+trim_x[p+1])/2)*spx*br*1.025;
end;
jumlah_wedge1 := 0.00;
for p := 1 to nb do
jumlah_wedge1 := jumlah_wedge1+wedge1[p];
for p := 1 to nb do
if p < 11 then
begin
   abot1[p] := (trim_x[p+1]*spx*br*1.025);
   abot2[p] := (trim_x[p]-trim_x[p+1])*(spx/2)*br*1.025;
end else
begin
   abot1[p] := (trim_x[p]*spx*br*1.025);
   abot2[p] := (trim_x[p+1]-trim_x[p])*(spx/2)*br*1.025;
end;
for p := 1 to nb do abot[p] := abot1[p]+abot2[p];
jumlah_abot := 0.00;
for p := 1 to nb do jumlah_abot := jumlah_abot+abot[p];
for p := 1 to nb do momen[p] := abot2[p]*(spx/6);
for p := 1 to nb do ttbrt[p] := momen[p]/abot[p];
for p := 1 to nb do
if p < 11 then
begin
   lng[p] := arm[p]+ttbrt[p];
end else
begin
   lng[p] := arm[p]-ttbrt[p];
end;
momen_baji := 0.00;
for p := 1 to nb do
momen_baji := momen_baji+(abs(abot[p])*lng[p]);

for p := 1 to 9 do begin
distr1[p] := abot[p]*(0.5+(((spx/2)+ttbrt[p])/spx));
distr2[p] := abot[p]*(0.5-(((spx/2)+ttbrt[p])/spx));
end;
distr1[10] := abot[10]*(0.5+(((spx/2)-ttbrt[10])/spx));
distr2[10] := abot[10]*(0.5-(((spx/2)-ttbrt[10])/spx));
```

```
    for p := 20 downto 12 do begin
    distr1[p] := abot[p]*(0.5+(((spx/2)+ttbrt[p])/spx));
    distr2[p] := abot[p]*(0.5-(((spx/2)+ttbrt[p])/spx));
    end;
    distr1[11] := abot[11]*(0.5+(((spx/2)-ttbrt[11])/spx));
    distr2[11] := abot[11]*(0.5-(((spx/2)-ttbrt[11])/spx));
    distr[1] := distr1[1];
    for p := 2 to 8 do
    distr[p] := distr2[p-1]+distr1[p];
    distr[9] := distr2[8]+distr1[9]+distr2[10];
    distr[10] := distr2[9]+distr1[10];
    distr[20] := distr1[20];
    for p := 19 downto 13 do
    distr[p] := distr2[p+1]+distr1[p];
    distr[12] := distr2[13]+distr1[12]+distr2[11];
    distr[11] := distr2[12]+distr1[11];

    jumlah_baji := 0.00;
    for p := 1 to nb do jumlah_baji := jumlah_baji+distr[p];
    if (abs(dlcg-lcg) > 0.0001) then
    begin
       for p := 1 to nb do cor_bod[p] := bod[p]+distr[p];
    end else
    begin
       for p := 1 to nb do cor_bod[p] := bod[p];
    end;
    bouyancy := 0.00;
    for p := 1 to 10 do bouyancy := bouyancy+(distr[p]*arm[p]);
    bou := bouyancy*2;
    b_tx := 0.00;
    for p := 1 to nb do b_tx := b_tx+cor_bod[p];
    b_dt := 0.00;
    for p := 1 to nb do b_dt := b_dt+(arm[p]*cor_bod[p]);
    lcb := b_dt/b_tx;
end;

procedure loading3;
var
  p : integer;
begin
    for p := 1 to nb do
    begin
       load_dat[p] := (wedge[p]-shipw[p]);
    end;
end;

procedure shearing;
var
  p,ns   : integer;
```

```
  s_dt   : data;
begin
    ns := nb+1;
    s_dt[1] := load_dat[1]*spx;
    for p := 2 to nb do
    s_dt[p] := s_dt[p-1]+(load_dat[p]*spx);
    for p := 1 to nb do
    shear_dat[p] := s_dt[p];
end;

procedure bending;
var
  p,ns  : integer;
  s_dt  : data;
begin
    ns := nb+1;
    s_dt[1] := 0.00+((0.00+shear_dat[1])/2)*spx;
    for p := 2 to nb do
    s_dt[p] := s_dt[p-1]+(((shear_dat[p-1]+shear_dat[p])/2)*spx);
    for p := 1 to nb do bend_dat[p] := s_dt[p];
end;

procedure sloping;
var
  p,ns              : integer;
  s_dt,m_dt         : data;
begin
    ns := nb+1;
    for p := 1 to nb do m_dt[p] := bend_dat[p]/(ine[p]*young);
    s_dt[1] := 0.00+((0.00+m_dt[1])/2)*spx;
    for p := 2 to nb do
    s_dt[p] := s_dt[p-1]+(((m_dt[p-1]+m_dt[p])/2)*spx);
    for p := 1 to nb do slope_dat[p] := s_dt[p];
end;

procedure curving;
var
  p,ns       : integer;
  s_dt       : data;
begin
    ns := nb+1;
    s_dt[1] := 0.00+((0.00+slope_dat[1])/2)*spx;
    for p := 2 to nb do
    s_dt[p] := s_dt[p-1]+(((slope_dat[p-1]+slope_dat[p])/2)*spx);
    for p := 1 to nb do cfd_dat[p] := s_dt[p];
end;

procedure dock_cur;
var
```

```
  dat1,dat2,dat3,dat4  : data;
begin
    loading2(nb,wd,bad,bod,wedge,dlcg,spx,br,beam,dat1);
    loading(nb,wd,bad,dat1,wedge,dat2);
    shearing(nb,spx,dat2,dat3);
    bending(nb,spx,dat3,dat4);
    sloping(nb,spx,dat4,ine1,young1,cfs);
    curving(nb,spx,cfs,cfd);
end;

procedure ship_cur;
var
  dat1,dat2,dat3,dat4  : data;
begin
    loading1(nb,ilcg,spx,ls,shipw,rb,dat1);
    loading3(nb,shipw,dat1,dat2);
    shearing(nb,spx,dat2,dat3);
    bending(nb,spx,dat3,dat4);
    sloping(nb,spx,dat4,ine2,young2,cfs);
    curving(nb,spx,cfs,cfd);
end;

procedure dock_cur2;
var
  dat2,dat3,dat4  : data;
begin
    loading(nb,wd,bad,cor_bod,wedge,dat2);
    shearing(nb,spx,dat2,dat3);
    bending(nb,spx,dat3,dat4);
    sloping(nb,spx,dat4,ine1,young1,cfs);
    curving(nb,spx,cfs,cfd);
end;

procedure ship_cur2;
var
  dat2,dat3,dat4  : data;
begin
    loading3(nb,shipw,wedge,dat2);
    shearing(nb,spx,dat2,dat3);
    bending(nb,spx,dat3,dat4);
    sloping(nb,spx,dat4,ine2,young2,cfs);
    curving(nb,spx,cfs,cfd);
end;

procedure make_mat1;
var
  kol,brs        : integer;
  dsp,dcur,temp_x  : data;
begin
```

```
    for kol := 1 to nb do
       temp_x[kol] := wedge[kol];
    for kol := 1 to nb do
    begin
        temp_x[kol] := wedge[kol]+delx[kol];
        dock_cur2(nb,wd,bad,cor_bod,temp_x,spx,ine1,young1,dsp,dcur);
        for brs := 1 to nb do
        begin
            mat1[brs,kol] := dcur[brs];
        end;
        temp_x[kol] := wedge[kol];
    end;
end;

procedure make_mat2;
var
  kol,brs        : integer;
  dsp,dcur,temp_x  : data;
begin
    for kol := 1 to nb do
       temp_x[kol] := wedge[kol];
    for kol := 1 to nb do
    begin
        temp_x[kol] := wedge[kol]+delx[kol];
        ship_cur2(nb,shipw,temp_x,spx,ine2,young2,dsp,dcur);
        for brs := 1 to nb do
        begin
            mat2[brs,kol] := dcur[brs];
        end;
        temp_x[kol] := wedge[kol];
    end;
end;

procedure mat_maker1;
var
  k,b : integer;
begin
    for k := 1 to nb do
    begin
        for b := 1 to nb do mat_x1[b,k] := dat1[b]
    end;
end;

procedure mat_maker2;
var
  k,b : integer;
begin
    for k := 1 to nb do
    begin
```

```
        for b := 1 to nb do mat_x2[b,k] := dat2[b]
    end;
end;

procedure jacobian;
var
  k,b     : integer;
begin
    for k := 1 to nb do
    begin
        for b := 1 to nb do
        jac[b,k] := ((mat1[b,k]-dat1[b])/delx[b])-((mat2[b,k]-dat2[b])/delx[b]);
    end;
end;

end.
```

PROSEDUR LU DECOMPOSITION

```
unit Dekomposisi_cad;

interface
uses prosedur2_cad;

procedure ludcmp(var a : matrix;
                    n : integer;
                 var indx : integ;
                 var d : real);
procedure lubksb(var a : matrix;
                    n : integer;
                 var indx : integ;
                 var b : data);
procedure invers(n   : integer;
                 a   : matrix;
                 var y   : matrix);

implementation

procedure ludcmp(var a : matrix;
                    n : integer;
                 var indx : integ;
                 var d : real);
const
    tiny = 1.0e-20;
var
  k,j,imax,i   : integer;
  sum,dum,big  : real;
  vv           : ^data;
begin
    new(vv);
    d := 1.0;
    for i := 1 to n do begin
       big := 0.0;
       for j := 1 to n do
          if abs(a[i,j])>big then big := abs(a[i,j]);
       if big = 0.0 then begin
        writeln('pause in Ludcmp - singular matrix');
        readln;
       end;
       vv^[i] := 1.0/big;
    end;
    for j := 1 to n do begin
       for i := 1 to j-1 do begin
          sum := a[i,j];
          for k := 1 to i-1 do
```

```
            sum := sum-a[i,k]*a[k,j];
          a[i,j] := sum
        end;
        big := 0.0;
        for i := j to n do begin
          sum := a[i,j];
          for k := 1 to j-1 do
            sum := sum-a[i,k]*a[k,j];
          a[i,j] := sum;
          dum := vv^[i]*abs(sum);
          if dum >= big then begin
            big := dum;
            imax := i;
          end
        end;
        if j <> imax then begin
          for k := 1 to n do begin
            dum := a[imax,k];
            a[imax,k] := a[j,k];
            a[j,k] := dum
          end;
          d := -d;
          vv^[imax] := vv^[j]
        end;
        indx[j] := imax;
        if a[j,j] = 0.0 then a[j,j] := tiny;
        if j <> n then begin
          dum := 1.0/a[j,j];
          for i := j+1 to n do
          a[i,j] := a[i,j]*dum
        end
    end;
    dispose(vv);
end;

procedure lubksb(var a    : matrix;
                     n    : integer;
                 var indx : integ;
                 var b    : data);
var
  j,ip,ii,i : integer;
  sum : real;
begin
    ii := 0;
    for i := 1 to n do begin
      ip := indx[i];
      sum := b[ip];
      b[ip] := b[i];
      if ii <> 0 then
```

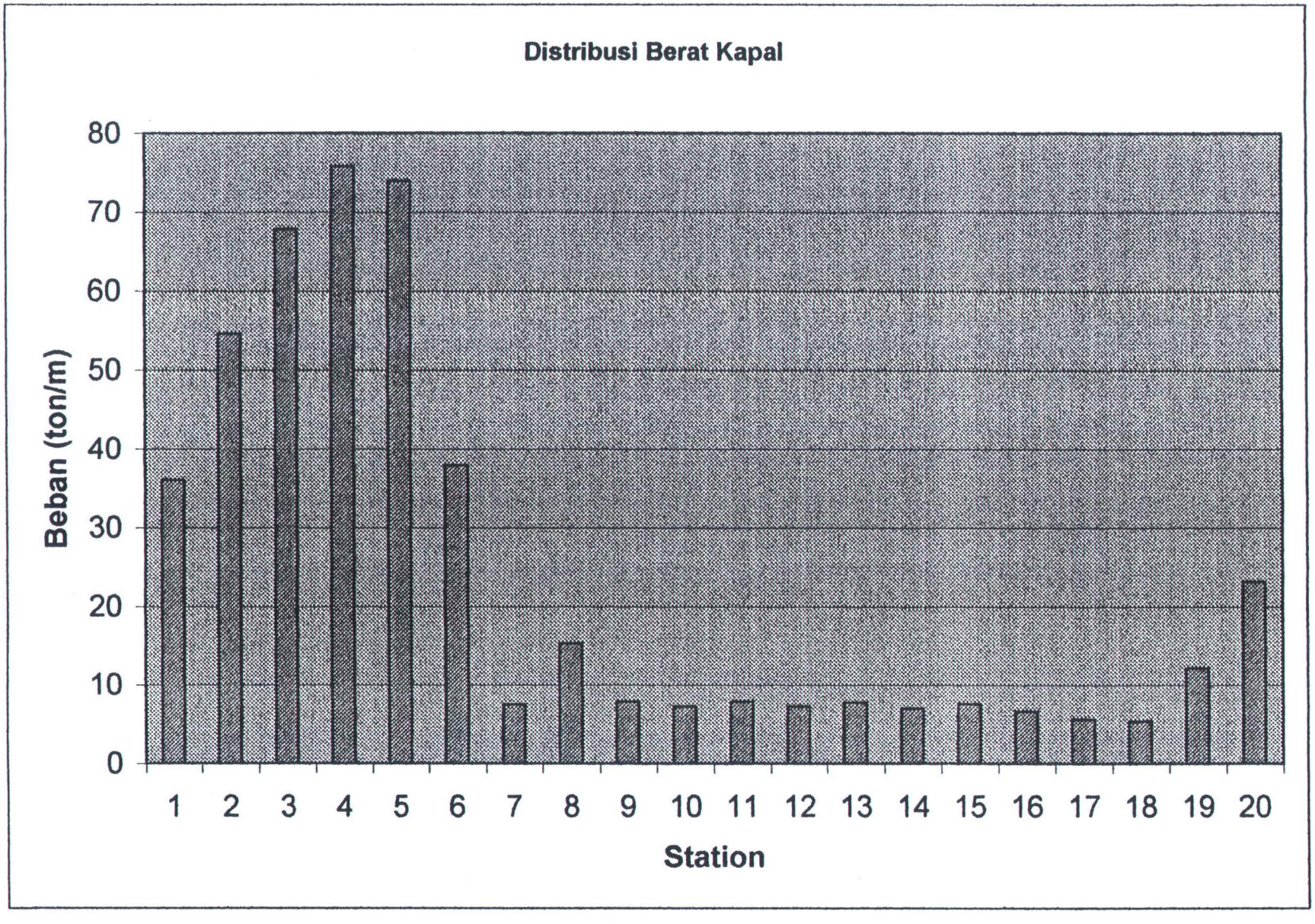
```
            ran3ma[i] := ran3ma[i]-ran3ma[1+((i+30) mod 55)];
            if ran3ma[i] < mz then
            ran3ma[i] := ran3ma[i]+mbig
        end
    end;
    ran3inext := 0;
    ran3inextp := 31;
    idum := 1
  end;
  ran3inext := ran3inext+1;
  if ran3inext = 56 then
  ran3inext := 1;
  ran3inextp := ran3inextp+1;
  if ran3inextp = 56 then ran3inextp := 1;
  mj := ran3ma[ran3inext]-ran3ma[ran3inextp];
  if mj < mz then mj := mj+mbig;
  ran3ma[ran3inext] := mj;
  ran3 := mj*fac
end;

function irbit1(var iseed : integer): integer;
const
  ib1 = 1; ib3 = 4;
  ib5 = 16; ib14 = 8192;
var
  newbit : boolean;
begin
  newbit := (iseed and ib14) <> 0;
  if iseed and ib5 <> 0 then newbit := not newbit;
  if iseed and ib3 <> 0 then newbit := not newbit;
  if iseed and ib1 <> 0 then newbit := not newbit;
  iseed := iseed shl 1;
  if newbit then
  begin
    irbit1 := 1;
    iseed := iseed or ib1
  end else
  irbit1 := 0
end;

procedure metro(KeelReaksi,t      : real;
          var ans          : boolean);
begin
  metropjdum := -1;
  ans := (KeelReaksi < 0.0) or (ran3(metropjdum) < exp(-KeelReaksi/t))
end;

end.
```

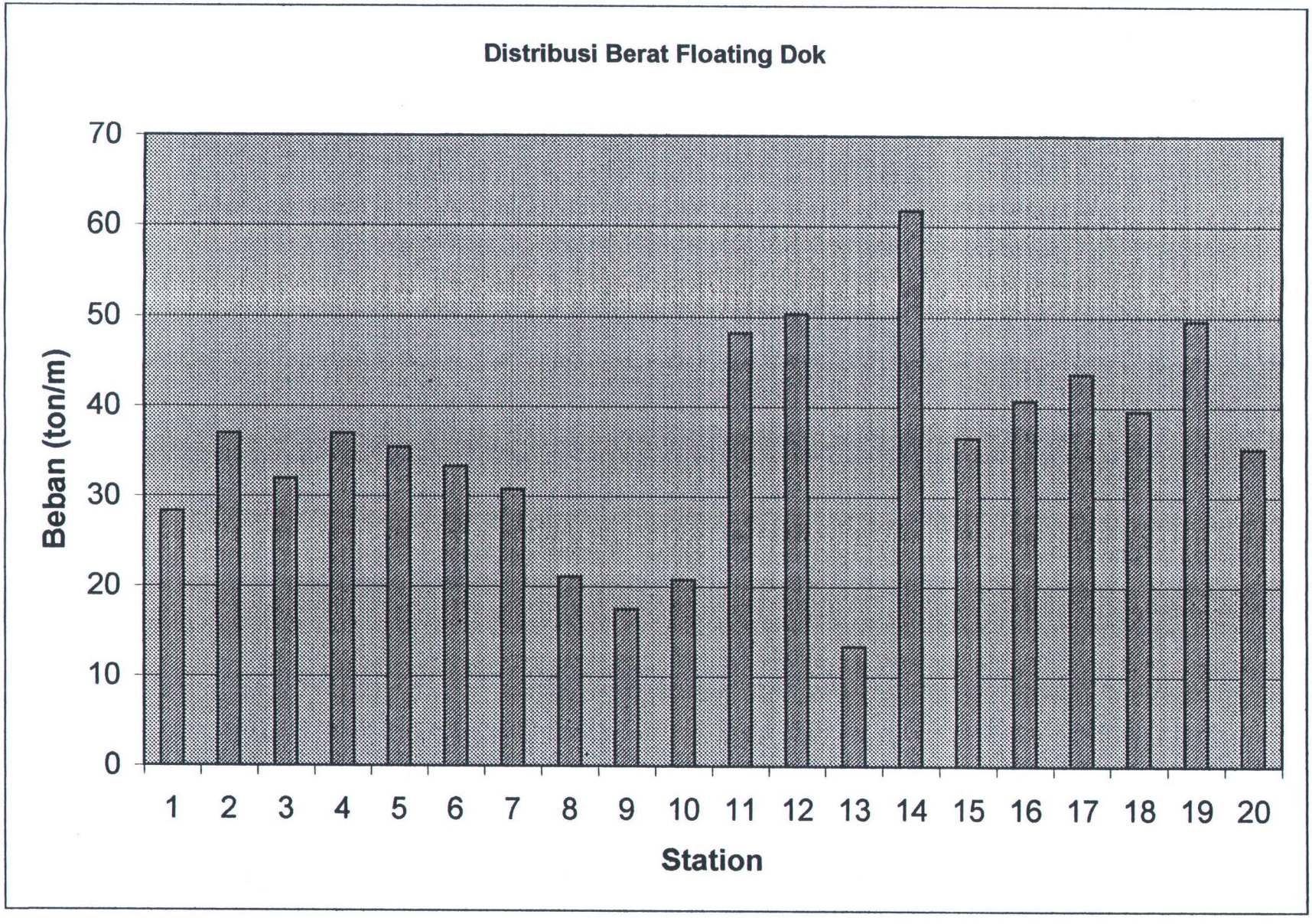
```
    end;
    closefile(fz);
end;

procedure TFormOptimasi.Simpan;
var
  a,p   : integer;
begin
    sopo := 'Hasil Optimasi.txt';
    assignfile(hasil,sopo);
    rewrite(hasil);
    for a := 1 to r do begin
    writeln(hasil,'Selisih Defleksi : ',ite_dat.nilai[a]:5:15);
    end;
    writeln(hasil,'Calculation terminated after : ',r:3,' iterations');
    writeln(hasil,'Delta when terminated        : ',tester:20:15);
    writeln(hasil,'Total keel block reactions   : ',lwtship:10:18);
    for p := 1 to nkeel do begin
    writeln(hasil,'Reaksi keel blok pada station ',p,' : ',
    blocks[p]:5:15,' [t/m]');
    end;
    for p := 1 to nkeel do begin
    writeln(hasil,'ballast di ',p,' : ',dock_dat.ba_dat[p]:5:15);
    end;
    for p := 1 to nkeel do vdelta[p]:=abs(def_1[p]-def_2[p]);
    writeln(hasil,'Station  Dock deflection  Ship deflection  Delta');
    for p := 1 to nkeel do begin
    writeln(hasil,p:length('Station'),def_1[p]:15,'  ',def_2[p]:15,'  ',
    vdelta[p]:15);
    end;
    closefile(hasil);
    if ioresult = 0 then
    begin
        PlaySound('c:\Win\Media\The Microsoft Sound.wav',0,snd_sync);
        if MessageDlg('File berhasil disimpan',mtConfirmation,[mbOK],0)
        = mrOK then begin
        close;
        end;
    end else
    begin
        if MessageDlg('File gagal disimpan !!',mtError,[mbOK],0) = mrOK
        then begin
        close;
        end;
    end;
end;

procedure TFormOptimasi.Acak;
label 10;
```

```
const
    min = 0.00;
    max = 3.60;
var
  tinggi1, sementara  : real;
  i                   : integer;
  vol1                : real;
begin
    10:
    Randomize;
    tinggi1 := random;
    if (tinggi1 < min) or (tinggi1 > max) then goto 10
    else vol1 := tinggi1*2.4*25.4;
    sementara := vol1/10;
    for i := 1 to 20 do ballast[i] := sementara;
    for i := 1 to nkeel do begin
        dock_dat.ba_dat[i] := ballast[i];
    end;
end;

procedure TFormOptimasi.Sukses;
var
  q   : integer;
  jumlah : real;
begin
    Acak;
    jumlah := 0.00;
    for q := 1 to nkeel do
    jumlah := jumlah + dock_dat.wd_dat[q];
    for q := 1 to nkeel do
    dock_dat.rs_dat[q] := (10987-jumlah-dock_dat.ba_dat[q])/100;
end;

procedure TFormOptimasi.Optimum;
var
  p,q,s,t,a     : integer;
begin
    r := 0;
    repeat
    if r > 0 then

    loading1(nkeel,inlcg,spur,bm,dock_dat.ws_dat,dock_dat.rs_dat,blocks);
    loading3(nkeel,dock_dat.ws_dat,blocks,blocks1);
    loading2(nkeel,dock_dat.wd_dat,dock_dat.ba_dat,dock_dat.bd_dat,blocks,
    lcgdock,spur,breadth,bm,co_bod);
    {t := r + 1;}
    for q := 1 to nkeel do xdel[q]:=(blocks[q])*1.0e-11-(0.0001*(r-1));

    dock_cur(nkeel,dock_dat.wd_dat,dock_dat.ba_dat,co_bod,blocks,lcgdock,
```

```
    spur,breadth,bm,dock_dat.in_dat,mdock,slop_1,def_1);
    ship_cur(nkeel,inlcg,spur,bm,dock_dat.ws_dat,dock_dat.rs_dat,blocks,
    dock_dat.i_dat,mship,slop_2,def_2);
    loading(nkeel,dock_dat.wd_dat,dock_dat.ba_dat,co_bod,blocks,blocks1);
    dock_cur2(nkeel,dock_dat.wd_dat,dock_dat.ba_dat,co_bod,blocks,span,
    dock_dat.in_dat,mdock,slop_1,def_1);
    ship_cur2(nkeel,dock_dat.ws_dat,blocks,span,dock_dat.i_dat,mship,
    slop_2,def_2);
    for q:= 1 to nkeel do vfunct[q,1]:=def_1[q]-def_2[q];

    make_mat1(nkeel,dock_dat.wd_dat,dock_dat.ba_dat,co_bod,blocks,span,
    dock_dat.in_dat,mdock,xdel,mat_c);
    make_mat2(nkeel,span,dock_dat.i_dat,mship,dock_dat.ws_dat,blocks,
    xdel,mat_d);
    mat_maker1(nkeel,def_1,mat_a);
    mat_maker2(nkeel,def_2,mat_b);
    matr_sub(nkeel,mat_a,mat_b,mat_e);
    jacobian(nkeel,mat_c,mat_d,def_1,def_2,xdel,jacob);
    Invers(nkeel,jacob,jacob_inv);

    matr_mul(nkeel,nkeel,1,jacob_inv,vfunct,jacob1);

    for q := 1 to nkeel do temp_blocks[q] := blocks[q];

    for q := 1 to nkeel do blocks[q] := temp_blocks[q]-(jacob1[q,1]);

    lwtship := 0.00;
    for q := 1 to nkeel do lwtship := lwtship + blocks[q];
    for q := 1 to nkeel do vdelta[q] := blocks[q] - temp_blocks[q];
    for q := 1 to nkeel do temp_blocks[q] := blocks[q];

    stopper(nkeel,vdelta,tester);

    r := r + 1;
    ite_dat.nx[r] := r;
    ite_dat.nilai[r] := tester;

    for q := 1 to nkeel do dock_dat.rs_dat[q] := temp_blocks[q];

    until r = 20;
    {(abs(tester) <= 0.00000000001);}

    KeelReaksi := 0;
    for q := 1 to nkeel do
    begin
       KeelReaksi := KeelReaksi + sqr(def_2[q]);
    end;
end;
```

```
procedure TFormOptimasi.Reaksi(Sender: TObject);
const tfactr = 0.09;
var nsucc,idum   : integer;
   nover,nlimit : integer;
   j,k          : integer;
   t            : real;
   ans          : boolean;
begin
    nkeel := 20;
    bm := 100.00;
    breadth := 26.00;
    inlcg := 8.5214827296;
    lcgdock := 0.00;
    buka;
    spur := bm/nkeel;
    span := bm/(nkeel-1);
    mdock := 2e11; mship := 2e11;

    nover := 1*nkeel;
    nlimit := 1*nkeel;
    idum := -1;
    t := 0.5;

    for j := 1 to 25 do
    begin
        nsucc := 0;
        k := 0;
        repeat
            k := k + 1;
            Optimum;
            metro(KeelReaksi,t,ans);
            if ans then
            begin
                nsucc := nsucc+1;
                Sukses;
            end else begin
                Sukses;
            end;
        until (nsucc >= nlimit) or (k >= nover);
    end;
    t := t * tfactr;
    for j := 1 to nkeel do begin
    Defleksi[j] := ((j/nkeel)*def_1[nkeel])-def_1[j];
    end;
    Simpan;
end;

end.
```


Distribusi Berat Kapal
Beban (ton/m)
0
10
20
30
40
50
60
70
80
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Station


Distribusi Berat Floating Dok
Beban (ton/m)
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Station